Definition and Configuration of Reliable Event Detection for Application in Wireless Sensor Networks

Von der Fakultät für Mathematik, Naturwissenschaften und Informatik
der Brandenburgischen Technischen Universität Cottbus

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation

vorgelegt von

Diplom-Informatiker

Steffen Ortmann

Geboren am 30.10.1980 in Eisenhüttenstadt

Gutachter: Prof. Dr. rer. nat. Peter Langendörfer

Gutachter: Prof. Dr.-Ing. Jörg Nolte

Gutachter: Prof. Dr. Torsten Braun

Tag der mündlichen Prüfung: 06. Juli 2010

To my family

Contents

Extended Abstract

Ubiquitous systems based on wireless sensor networks will amazingly increase our quality of life. These systems are to be deployed in large areas with high density where hundreds or thousands of nodes are used. Certainly that demands to use low cost devices with limited resources, which in turn are prone to faulty behaviour. This work introduces a novel concept for wireless sensor network configuration considering fault tolerance, energy efficiency and convenience as primary goals while being tailored to user needs. It allows to ignore low-level details like node resources, network structures, node availability etc. and enables the programmer to work on a high abstraction level, namely the event itself including event related constraints. The definition of events characterising real world phenomena is of prominent use in sensor networks. The presented concept autonomously configures and monitors events, even if it requires to organise collaboration between nodes to deliver the results.

The contribution of this work is threefold. An intuitive XML-based Event Specification Language (ESL) simplifies event configuration to a level that is even suitable for non-professionals. It features hardware independent description elements to define complex phenomena and enhances these by tailor-made voting schemes and application constraints. Based on that, a novel, fully decentralised mechanism to autonomously set up distributed event detection called Event Decision Tree (EDT) and a cost efficient means to maintain such EDT, are presented. EDTs can be efficiently constructed on every device by using a tiny generating finite state machine requiring eight states only. It enables every node to self-divide event queries according to its own resources and self-adapt to the tasks assigned. Simultaneously, the EDT provides the interface for efficient collaboration using a lease-based publish/subscribe approach. The simulations clearly show that this concept works well and the applied collaboration scheme outperforms even idealised acknowledgement-based approaches.

On top of the EDT, a means is developed that enhances the reliability of detection beyond the scope of Boolean event decision. It examines behavioural trends in sensor readings to indicate the significance of actual measurements in relation to the configured event. Measured data is investigated in detail to finally attach a significance indicator i_S to each event. This automatically generated indicator shall support users or overlaying systems in decision-making. In the example scenario based on data of real test cases, the i_S indicates a flaming fire 88 seconds and a smouldering fire 48 seconds before the threshold-based method triggers the alarm.

Kurzfassung

Drahtlose Sensornetze stellen eine Informationstechnologie dar, deren ubiquitäre Verwendung unsere zukünftige Lebensweise entscheidend beeinflussen und verbessern kann. Der massenhafte Einsatz solcher Systeme bedingt strengste Einschränkungen in Kosten und Ressourcen und führt damit zu einer verringerten Zuverlässigkeit der einzelnen Sensorknoten und des gesamten Sensornetzes. Diese Arbeit führt ein neuartiges Konzept zur Konfiguration von drahtlosen Sensornetzen ein. Die Schwerpunkte liegen dabei auf Fehlertoleranz und Energieeffizienz unter besonderer Berück-sichtigung von Verbraucherfreundlichkeit und Komplexität. Es erlaubt dem Nutzer auf einem hohen Abstraktionsniveau zu arbeiten, ohne niedrigstufige Details wie Sensor-Ressourcen, Netzwerkstrukturen, Verfügbarkeit der Knoten usw. beachten zu müssen. Die Beschreibung physikalischer Phänomene als Events findet breite Anwendung bei der Programmierung von Sensornetzen. Das vorgestellte Konzept konfiguriert und beobachtet einmalig definierte Events vollautomatisch, auch wenn dazu kooperative Beziehungen zwischen verschiedenen Sensorknoten notwendig sind.

Die Arbeit besteht aus 3 Hauptteilen. Eine intuitive auf XML basierende Sprache zur Definition von Events vereinfacht die Konfiguration der Sensornetze auf ein Niveau, das auch für konventionelle Nutzer angemessen ist. Es besitzt hardware-unabhängige Elemente zur Definition von komplexen Phänomenen (Events) und erweitert diese durch maßgeschneiderte Verfahren zum Mehrheitsentscheid (Voting) und verfeinerte Ausführungsbedingungen. Darauf aufbauend wird ein neuartiger, komplett dezentraler Mechanismus zur autonomen verteilten Event Erkennung, genannt Event Entscheidungsbaum (Event Decision Tree (EDT)), vorgestellt. EDTs werden auf sehr effiziente Weise auch auf kleinsten Geräten durch einen generierenden endlichen Automaten mit lediglich acht Zuständen erstellt und verwaltet. Der EDT ermöglicht es jedem Sensorknoten, sich auf Grundlage seiner verfügbaren Ressourcen die übermittelten Aufgaben selbstständig einzuteilen und zu konfigurieren. Gleichzeitig bildet er die Schnittstelle für effiziente Zusammenarbeit zwischen mehreren Knoten mittels eines Lease-basierten Publish/Subscribe Verfahrens. Die Aufwandsschätzungen und Simulationen zeigen deutlich, dass dieses Verfahren sehr effizient arbeitet und sogar idealisierte auf Bestätigungen (Acknowledgements) basierende Verfahren deklassiert.

Zusätzlich wurde ein Verfahren entwickelt und getestet, das die Zuverlässigkeit in der Erkennung von Events über die üblichen, auf Schwellwerten basierenden, Methoden hinaus steigern kann. Es untersucht den Trend vergangener Messwerte um die Relevanz aktueller Messwerte in Relation zum Event zu ermitteln. Daraus ergibt sich ein Indikator für die Signifikanz des Messwerts (i_S). Dieser automatisch generierbare Indikator soll Nutzer oder höherrangige Systeme bei Entscheidungen und weiteren Aktionen unterstützen. In der vorgestellten Beispielanwendung, die auf Daten aus realen Testläufen basiert, indiziert i_S ein flammendes

Feuer 88 Sekunden und einen Schwelbrand 48 Sekunden vor den auf Schwellwerten basierten Methoden, die erst entsprechend später einen Alarm auslösen.

Acknowledgements

Finishing this dissertation would have been impossible without the help of many persons. I would like to express my gratitude to all of them.

First, I would like to thank my supervisor Prof. Dr. Peter Langendörfer for accepting me as his PhD student. He took care about all technical and non-technical stuff that was necessary to finish this thesis successfully. He always listened to my problems and regularly proofread this work. By that, he provided valuable hints, criticism and encouragements. On this note, I would also like to thank Prof. Dr. Rolf Kraemer. His support allowed me to go my own ways in research.

Second, I would like to spend a big thank to my colleague Michael Maaser for supporting this work with a lot of ideas and hints. He read and proofread this thesis again and again and again. Finish this thesis in time would not have been possible without his assistance.

Next, I would like to thank my wife and my family for their support during all the time. They really cared about me and tolerated my sometimes strange behaviour. They made sure that I stayed focused and gave me the necessary freedom to work. Without their encouragement and understanding it would have been impossible to finish this work.

Further, I would like to thank all of my colleagues of the system department of IHP for their advice and assistance. We had a very nice time and it has been a privilege to work with them.

Last but not least, I would like to thank Prof. Dr. Theodor Vierhaus and all the other organisers of the International Graduate School of Cottbus for financing this work by a PhD scholarship.

Chapter 1

Introduction

Pervasive computing significantly increases the human-computer as well as the environment-computer interaction and enables a direct interplay between the real world and the information technology. The recent advances in the areas of semiconductor industry and computer sciences indicate that the vision of pervasive intelligent environments surrounding and serving us at any place and any time [72], is becoming reality in the near future. Computing devices will be embedded in everyday objects, e.g., in coffee cups [21], allowing information technology to fade into the background and become nearly invisible to their users. As one of the first real world examples enabling pervasive computing, Wireless Sensor Networks (WSNs) have become a rising star in this research field. Envisioned to be distributed like "Smart Dust" [29, 30], these networks support a broad range of applications [1, 2] and may become the perfect service and surveillance tool [9]. Based on their capabilities to identify physical phenomena, sensor networks can be applied for environmental and structural control [16, 40, 64, 73], context-awareness for personal services [57], military applications [22] and ubiquitous healthcare [19], to mention a few. To summarise, ambient assisting technology based on WSNs will amazingly increase our quality of life.

Despite of the emerging advantages and potential applications, there are still a lot of challenges and problems to solve before WSNs can be also used as consumer technology. WSNs are expected to be deployed in large areas with high density where hundreds or thousands of nodes are used. Certainly that demands to use low cost devices with limited resources, which in turn are prone to faulty behavior. This thesis is focusing on means to enhance the reliability of Wireless Sensor Network (WSN) without asking for improved device quality. The Federal Standard 1037C [18] of the United States defines the term fault as:

An accidental condition that causes a functional unit to fail to perform its required function.

Due to the pervasiveness of envisioned systems, those are caught in a crossfire of external and internal influences that increase the fault probability. Sudden changes in operational conditions, varying deployment and hazardous environments adversely affect the reliability of application. To make matters worse, sensor nodes are subject to strict energy constraints providing limited power only. Thus, inexpensive fault tolerant sensor networks that achieve a high reliability while remaining energy efficient are in great demand.

Traditional sensor network applications report all sensor readings to a global sink either continuously or if certain conditions are met. Sinks are usually special nodes that provide more resources and make the final decision about sensed phenomena based on received readings. Such data gathering applications exchange huge amounts of data, cause much traffic, consume much energy and hence, reduce lifetime, throughput and responsiveness of the network. Thus, only certain changes in sensor readings, called events, are transmitted. Events provide a suitable abstraction of real world phenomena [59], whose physical properties can be measured by sensors. Events typically describe a number of measurement related constraints, e.g., thresholds of sensor readings. Sensors fire an event if current measurements indicate the exceedance of these thresholds. Fired events usually trigger further actions, such as the activation of alarms or the recording of detailed data for further analysis.

There are primitive and complex events. Primitive events describe the exceedance of one configured threshold by a single sensed value. Many applications demand detecting the simultaneous occurrence of several primitive events, particularly if identification of complex real-world phenomena is required. A combination of several primitive events is a complex event. For example, the occurrence of an event *fire* should be denoted as a combination of the primitive events *(temperature > 80℃) AND (smoke ≥ 1,1%)* instead of using the primitive events only. Complex events based on different sensing capabilities indicating the same phenomena, here temperature and smoke, gain a higher false alarm safety and enhance the reliability of event detection [62]. Efficient information-fusion for complex events is already a challenge for single devices, but gets even harder if the sensor nodes do not provide all sensing capabilities needed for event detection. In that case, sensor nodes must collaborate and share their sensing capabilities to continue with event detection. For reliable application it is a necessity to enable sensor nodes to autonomously deal with different conditions as being expected in pervasive systems, i.e., heterogeneous distributed sensing capabilities, node mobility, varying network topology, failed sensors or sensing units etc.

Available WSNs face two major problems. These are high fault probability and configuration complexity. There already exist fault tolerant techniques that can enhance the detection accuracy and the reliability of event-based applications in WSNs. Of course, they introduce a certain overhead. Besides improvements in

the fault tolerant performance, novel detection and collaboration techniques can significantly reduce the overhead associated to such techniques. In particular, the cost-efficiency of fault tolerance means is of interest, which determines the cost-benefit ratio between the enhancement of detection accuracy and the required overhead. That is considered necessary to enable reliable low power applications as required for long time deployments of WSNs. The contributions of this thesis are fault tolerant means that improve the fault tolerance of WSN-based applications while providing a high cost-efficiency. These means can be applied without the need of improved sensing devices.

There is further a lack of means to provide an ease of use for the definition and configuration of reliable event-based applications in WSNs. It is an obvious fact that most approaches for WSNs sparely consider or even disregard the complexity of configuring a WSN. However, a proper usability is considered essential for WSNs supporting real life applications. Making the programming and deployment of a WSN accessible for non experts could become the most important issue in order to make them widely accepted. To gain a broad consumer acceptance of WSNs, provision of means, that enable non-professional users to make use of the WSN is required. These users are usually short on experience of programming languages and sensor networks. The contribution of this thesis is a straightforward method for event-based task definition and sensor node configuration without requiring knowledge about hard- or software or node deployment.

Structure of this thesis The next Chapter presents related work and points out respective advantages and drawbacks. The major approaches are validated and compared with regard to design criteria for reliable application of WSN, which are also introduced. Chapter 3 gives an overview of the system architecture and introduces the Event Specification Language (ESL) for high level event-based task assignment and sensor programming. In addition, a central example scenario is provided. Based on that, Chapter 4 presents how to autonomously configure sensor nodes to event definitions by adaptive pruning of the Event Decision Tree (EDT). The efficiency of the applied publish/subscribe collaboration scheme is analysed and compared to an idealised acknowledgement (ACK)-based approach. The simulation results presented at the end of this Chapter, confirm the advantages of the publish/subscribe scheme. As an additional feature of using EDTs, Chapter 5 introduces an indicator of the significance of events i_S and evaluates its performance. Finally this work examines its contributions and concludes with a summary and an outlook on future work.

Chapter 2

Related Work

This Chapter introduces design criteria for reliable event-based detection schemes in WSNs. Considering these criteria, it presents and points out the basics of published approaches in three directions, the event definition, fault tolerant information-fusion and decentralised collaborative event detection. Finally, the major approaches are compared with respect to the introduced design criteria for reliable event-based detection schemes. This motivates the introduction of a novel concept for integrated event definition and node configuration. On top of this basic concept, a means to enhance the quality of event detection is announced. Respective related work in terms of Quality of Information (QoI) is separately provided in the last Section.

2.1 Design criteria for reliable event-based applications

In order to enable comparison of presented approaches and to set the objectives this work is aimed at, this section introduces design criteria for development of reliable event-based applications in WSNs. A suitable approach for reliable event detection in WSN must consider the following design criteria:

Fault tolerance This is the ability to detect different types of faults as well as means to correct those. Proper evaluation of both redundant and diversified data readings from different sources, enhances the reliability of event detection, e.g., by overruling incorrect results based on a majority decision. In ideal case, the overhead associated with fault tolerant methods is adaptable to the application it is used for. Critical tasks may demand a highly reliable and fault tolerant behavior while accepting additional overhead for those tasks. In contrast to that, low power applications such as climate control in vineyards, may discard all overhead to achieve very long node life cycles.

Adaptivity Sensor nodes and applications should provide means to continue event detection when the context changes, sensors fail or nodes move. The focus is on (re-)adapting on-node as well as in-network processing for automatic resource-oriented event configuration. This regards both, on-node adjustments in case of missing or failed sensing facilities and adaptation of distributed detection if relations to collaborating nodes are interrupted due to failed or moved nodes.

Autonomy In addition to the autonomous nature of sensor nodes, every node in the network must be enabled to perform all necessary tasks for event detection. A fully decentralised approach avoiding assignment of superior devices such as super nodes [13], event gateways [67] or fusion centres [71] prevents from having potential Single Point of Failures (SPoFs).

Transparency Dealing with heterogeneous nodes and network structures, sudden changes in the environment or failures during collaboration etc., consequently requires continuous adaptation and device configuration. These processes must be hidden to remain fully transparent to the user. Especially pervasive WSNs are expected to make use of various sensors with similar capabilities. An automatic hard- and software abstraction can cover such heterogeneity.

Energy efficiency Small devices like wireless sensor nodes usually are subject to strict energy constraints providing limited power only, e.g., by battery packs. Transmission is the most power-hungry operational mode of WSNs consuming orders of magnitudes more energy than local processing. Since collaboration simultaneously requires communication between sensor nodes, it significantly increases the energy consumption and hence, decreases the maximum reachable node lifetime. To cope with that, enhancing the cost-efficiency of collaboration by reducing the number of transmissions and the amount of exchanged data is of primary concern. In addition, all parameters regarding sensing intervals, duty and sleep cycles, adaptation rate etc., should ideally be configurable to best customise the energy consumption to application requirements.

Convenience It is an obvious fact that most approaches less consider or even disregard the complexity of configuring a WSNs. However, a proper usability is mandatory for WSNs supporting real life applications. To gain a broad consumer acceptance of WSN it is required to provide means that enable the non-professional users, who are usually short on experience of programming languages and sensor networks, to make use of a WSN. Therefore a straightforward method to define tasks and configure sensor nodes without requiring knowledge about hard- or software or node deployment is in demand.

2.2 High abstraction for sensor network configuration

Many higher abstractions for sensor network programming and configuration are already available. One of the most famous is the Declarative Database for Sensor Networks (TinyDB) [39] extending a Structured Query Language (SQL) to support in-network data queries on sensor nodes using the Tiny Operating System (TinyOS) [37]. Very similar to that is the The Sensor Network is the Database (COUGAR) project [76], which also supports data queries in a SELECT-FROM-WHERE SQL dialect. Both approaches provide a good abstraction layer to specify data collection in database-query style but still work on the node level. The supported data aggregation schemes also allow to fuse several readings to cover individual deviations in measurements. They further use a centralised topology with at least one coordinating node that continuously collects and evaluates raw sensor readings. COUGAR additionally addresses the possibility of node failures and specifies a second coordinating node that may help out if the elected coordinating node fails.

Out of available approaches, the Tiny Application Sensor Kit (TASK) [11] is most closely related to the aims of this work. It is a kit for configuring low data rate environmental monitoring applications while remaining "self-explanatory, easy to configure, and easy to maintain". It is based on TinyDB carrying mentioned drawbacks along but additionally provides a complete application background from field tools over gateways and internet connectivity up to support of external tools for proper sensor data preparation and network monitoring. In addition to the data collection features of TinyDB, TASK also supports inferior power management and considers fault tolerant performance. Despite experiences with huge discrepancies in sensing accuracy and calibration problems, fault tolerance is related to node crashes only. On detecting a failure, e.g., not sending or receiving messages anymore, a watchdog component restarts the node and retains state from nodes around it to continue data collection. It further deals with inaccurate sensor readings by reporting the actual value as the median of ten readings from the previous ten seconds. Altogether, TASK is a valuable step into the right direction but can still be improved in several points, particularly in managing energy efficiency and fault tolerance, e.g., enhancing the reliability of applications in case of uncertainty in sensor readings and failing sensing devices.

The macro programming language SpaceTime Oriented Programming (STOP) [68] creates data queries from a global viewpoint without considering details of single nodes. Based on migrating agents, who collect required data, STOP provides a more comfortable data collection but it requires a complex run time environment and virtual machines on every node. To summarise, all approaches transport huge amounts of data and analyse collected data by central nodes, which create a SPoF. They further do not allow the user to address and customise the necessary parameters regarding fault tolerant performance and energy

efficiency. Finally, these approaches still require to use scripting or programming languages, which is not feasible for non scientific deployment. A configuration concept that aims at ease of use for sensor configuration should be tailored to the user and self-configure to defined tasks. Thus, the user only needs to define what he is interested in, i.e., the description of the event to be sensed and event-related application constraints, without taking care of hardware, software and node deployment.

2.3 Fault tolerant voting mechanisms

Caused by low cost design or heavy noise, sensing devices attached to the sensor nodes may generate inaccurate or uncertain sensor readings, which increase the false alarm probability. Appropriate fusion of redundant information positively effects the mean time to failure [63]. Voting algorithms exploit redundant information in sensor networks, e.g., provided by multi-covered areas [12, 15], to compare readings of several nodes. The classic way of voting is Majority Voting (MV) [33] in which one node collects the results of neighbouring nodes and finally makes a majority decision about the correctness of received readings. Meanwhile, a number of derivative approaches have been published that differ in the applied voting algorithms as well as in how the region of event is established.

Krishnamachari et al. [35, 36] introduce a self-organising algorithm that provides a distributed fault tolerant approach for regional event extraction in sensor networks. All nodes with readings of interest in a proper neighbourhood, i.e., the region of event, are formed into a cluster where the node with the lowest id-number becomes the cluster head. The cluster head collects all readings and performs a MV. Since their approach is based on binary event detection where all nodes signal a “Yes” or “No” instead of sending their measurements, the cluster head simply counts all statements. If more than 50 percent of participating nodes state an event, the cluster head forwards this event to the central sink in the network. Beside performance issues, this approach does not take care on energy resources. Large event regions produce much overhead for communication and the election of the cluster head could be done more efficiently than by choosing the lowest id.

Luo et al. [38] presented a similar approach but exclusively consider the dissipation of energy during voting. It allows to choose a proper event region and to adapt the number of voting participants to a required level. An evaluation between detection accuracy and energy usage in comparison to the number of voters is presented. It clearly indicates that the efficiency of voting strongly depends on the application. Thus, the energy consumption of voting nodes must be adaptable to the required level of fault tolerance.

Sun et al. [64] establish Confidence Weighted Voting (CWV), which is based

on MV but in contrast to that, the CWV algorithm grants higher weights to sensor readings that are more likely to be correct. Each sensor node obtains a confidence value by comparing the sensing results in overlapping coverage areas. Sensors gain a higher confidence value if those frequently signal "correct" measurements compared to "faulty" results of neighbouring nodes. In other words, the confidence value of a sensor node increases if its sensor readings match the majority of all readings in the neighbourhood. The performance of CWV was compared to simple MV and Distance Weighted Voting (DWV), where the influence of other sensors decreases in distance to the voting initiator. The evaluation of CWV results in an increased resilience to sensor errors of at most 49 percent in contrast to the other voting schemes.

Krasniewski et al. proposed TIBFIT [34], a protocol for reliable and fault tolerant sensor networks that is able to cope with arbitrary data faults and malicious nodes. Similar to CWV, it assigns a trust value to the sensor nodes. These values confirm the plausibility of correct measurements or state a lack of credibility for single nodes. The head node of a voting region collects the readings and trust values of all nodes and decides whether an event has occurred or not. Due to the decision of the head node, the trust values of all correct reporting nodes increase whereas the other values decrease respectively. To make sure the trust values are correct, at least two shadow head nodes monitor all activities and results of the voting process in background and take corrective action if necessary. TIBFIT achieves a good fault tolerant performance even if more than 50 percent of the sensor measurements are faulty, provided that the monitoring phase is long enough to establish reliable trust values. Additionally this protocol is able to cope with malicious nodes, which can only temporary influence the voting because their confidence values decrease with every faulty report. This is only true if the number of malicious nodes is less than the number of correct reporting nodes. Unfortunately the required overhead was not measured or calculated, but the algorithms used for collecting and distributing the sensor readings and the trust values allow assumption of an enormous overhead, especially for the usage of shadow head nodes. Hence, the efficiency of the provided fault tolerant performance strongly depends on the application it is used for.

Voting is suitable to increase the reliability of sensor measurements indeed but available approaches lack of means to deal with varying network conditions and application requirements. Those approaches neither allow to adjust the number of voting participants nor to scale the size of voting regions. Hence, they are customisable for specific tasks but are inefficient if different events are to be evaluated. Likewise the problems of missing votes as well as required time for the voting process are not considered. Especially safety-critical applications demand to keep certain timing constraints and must adapt to changing conditions automatically. Further the dedicated overhead of fault tolerant methods such as

voting must be adaptable to the application to balance fault tolerant behavior against required overhead. The presented Event Specification Language (ESL) fine-tunes fault tolerant performance by combining the advantages of voting with self adapting approaches as well as with the benefits provided by heterogeneous sensing features [62].

Beside voting based techniques, there also exist a couple of different approaches to cope with uncertain detection results. Wang et al. [71] present a fault tolerant fusion role to combine several sensor measurements in local fusion centres by error-correction codes. It determines the minimum hamming distance between considered sensor data within certain vicinity. It represents a mixture of MV and data aggregation algorithms but assumes faulty sensor data to occur infrequently and isolated. Anomalous measurements that feature a certain hamming distance to the average data are considered as faulty. Those measurements are adjusted to the measurements of neighboured nodes. The fusion centres aggregate all adjusted measurements and forward the results to a central data sink. This approach has two main drawbacks. The fusion centre nodes that are responsible for a large cluster may smooth out locally detected events, which finally results in a non-identified event. This may happen if the phenomenon to be sensed features such a small volume expansion that it is only detected by a few or even single sensor nodes. By that, the detected event may be overruled by the other nodes in that fusion region. Further, applying fusion centres creates SPoFs and leaves no chance for task migration if these nodes fail.

Martincic et al. [42] establish event signatures as a distributed event detection scheme. They use position data to partition the sensor network into equally sized square cells where each sensor node is associated with a certain cell. These cells represent a matrix that contains the actual sensor readings. Each cell aggregates the readings of its assigned nodes by MV and distributes the result to neighbouring cells. Hence, each cell also holds a 3x3 matrix containing its own readings in the centre of this matrix and the aggregated sensor readings of the neighbouring cells as border entries. In this context, an event signature is a 3x3 event matrix determining the sensor readings that represent the phenomenon to be sensed. These event matrices are injected into the sensor network. If a cell detects a match between its local matrix and the injected event matrix, it signals an event and its location to the base station. The distribution of cell values automatically provides redundant data sources for fault tolerant data management and enables to cope with failed sensor nodes. The idea to inject event matrices further allows to reconfigure applications at runtime. By contrast, there are two main drawbacks. First, the applicability of this technique is significantly reduced because it does not allow to evaluate different sensor readings or several event matrices simultaneously. This limits the detection facilities to a certain kind of measurements and does not allow to detect complex phenomena that require the

simultaneous evaluation of several physical properties. Second, continuous transmission of cell values stresses the energy consumption of the sensor network.

For energy efficiency reasons, Nakamura et al. [43] proposed a reactive variant of event detection that assigns different roles to event related nodes. These nodes form into event clusters only if a reading of interest is detected to save energy otherwise. The head node aggregates all data and sends it to a central sink while using the shortest path with a maximum of inter- and intra-cluster data aggregation. Indeed, such a reactive algorithm can save enormous resources if events occur rarely but the efficiency surely depends on the application. Moreover, this approach only considers static events of fixed size and was simulated on a uniformly distributed sensor network, which is unrealistic for real applications.

2.4 Collaborative event detection

Another focus of this work is on providing reliable event detection even in case of missing resources, mobile nodes or failures in sensors and connectivity. Vu et al. [67] introduce a composite event detection scheme for sensor networks composed of different nodes with varying sensing capabilities. They split complex event detection among different nodes into sets of atomic events, which are similar to threshold values. Atomic events are merged by special gateway nodes to determine final results. The gateway nodes however build SPoFs. This approach provides configurable levels of fault tolerance by selecting an appropriate k for k-watching sets of sensors while considering the energy consumption and the event notification time but requires an expensive setup phase. Phani Kumar et al. [54] present a similar collaboration scheme. They create event-based trees for complex events containing all assigned sensor nodes. These nodes collaborate using a content-based publish/subscribe communication model, where child nodes publish readings of interest to parent nodes. The root node of the event tree obtains all sensor readings and decides about the monitored event. Again this root node is a SPoF and introduces some vulnerability to the system.

Kamiya et al. [31] apply a Peer-to-Peer (P2P) network of sensor gateways to maintain event detection across several heterogeneous sensor networks. Each sensor gateway accesses and manages a certain sensor network. To define event detection in one or several maintained sensor networks, the sensor gateways provide an eXtensible Markup Language (XML) event description parser that splits complex events into required atomic ones and registers these at the corresponding gateway nodes. The underlying sensor networks continuously report their raw sensor readings to the gateway nodes, which finally evaluate the atomic and respective complex events. Even here the sensor gateways constitute a SPoF. Just as all other discussed approaches relying on gateway or centralised nodes, this is again very inefficient with regard to energy consumption. Due to the

fact that atomic events are not forwarded down to the measuring sensor nodes for evaluation, all sensor readings need to be sent to the gateway nodes even in case of no event is triggered. By that the energy consumption is far from optimal. Nevertheless, autonomous management of sensor networks based on an event description parser is a promising approach to reduce the complexity of node and networks configuration. Unfortunately, the XML descriptions used at the parser are not presented what makes it impossible to draw conclusions about their applicability.

The Context Dependent Event Detection (CoDED) platform [61] presents an architecture for context-dependent event detection in sensor networks. In order to save energy resources, events are monitored in certain monitoring context only. The context description is defined by a propositional logic, which evaluates to true as long as a specified context is given. Combinations of primitive events may form global (and maybe distributed) complex events, which are observed by a composite event detection engine. That engine seems to adapt automatically to current network situations but the general question of how to distribute and process composite events on several devices is left open. Unfortunately, the authors do not provide implementation details.

2.5 Assessment of related work

For final comparison, the presented approaches are evaluated against the introduced criteria in Table 2.1. First of all, there exists no approach that addresses and fulfils all criteria. Krishnamachari [35, 36] and Krasniewski [34] provide the highest fault tolerance and even enable to cope with malicious nodes but increase the required overhead by at least a factor of three. Since it is a prerequisite to ensure reliability of detection, all approaches allow to adapt to changing conditions but usually focus on certain changes only, e.g., faults, malicious nodes, unavailable resources, environmental changes, connectivity etc. Except for CoDED [61], which unfortunately was not implemented so far, all approaches carelessly neglect the autonomy a sensor network requires. Here, fully distributed concepts not depending on special nodes are in great demand, which by design enables all nodes to fulfil every task required for event detection to prevent from SPoFs. TinyDB [39], COUGAR [76] and STOP [68] reduce the configuration complexity by high programming abstraction and transparency at the cost of autonomy and energy efficiency. Slightly better, TASK [11] also supports users with a lot of tools on top of improving event detection only.

	TinyDB	COUGAR	STOP	TASK	Krishnam.	Luo	Sun	Krasn.	Vu	P. Kumar	Kamiya	CoDED
Fault tolerance	(- -)	(0)	(-)	(0)	(++)	(+)	(+)	(++)	(+)	(+)	(0)	(0)
Adaptivity	(+)	(+)	(+)	(+)	(+)	(+)	(+)	(+)	(+)	(+)	(+)	(++)
Autonomy	(-)	(0)	(0)	(-)	(-)	(0)	(-)	(0)	(-)	(-)	(- -)	(+)
Transparency	(+)	(+)	(++)	(++)	(+)	(?)	(0)	(0)	(0)	(+)	(+)	(0)
Energy efficiency	(- -)	(0)	(-)	(-)	(- -)	(-)	(-)	(- -)	(0)	(- -)	(- -)	(-)
Convenience	(0)	(0)	(+)	(+)	(-)	(?)	(?)	(?)	(?)	(?)	(0)	(-)
Validation: (++) very good; (+) good; (0) regular; (-) bad; (- -) very bad; (?) not mentioned												

Table 2.1: Comparison of related work in consideration of the introduced design criteria. There exists no approach that addresses and fulfils all criteria. Most provided is fault tolerance, adaptivity and transparency whereas energy efficiency, autonomy and convenience are marginally taken into account or even are completely missing.

To summarise, most provided are fault tolerance and adaptivity whereas energy efficiency, autonomy and convenience are marginally taken into account or even are completely missing. As shown in the existing approaches, providing high fault tolerance is possible indeed but partially results in a significantly increased stress of resources. Particularly with regard to the energy consumption and cost-efficiency for collaboration, all approaches constantly perform poor or even bad. There is no doubt that fault tolerance requires an overhead but the efficiency of an application significantly depends on a proper balance between the application requirements and the implementation. Since the application defines the configuration requirements, a suitable approach must be customisable to these requirements and not vice versa. Further, most approaches shift final decisions to centralised nodes and are vulnerable if these nodes are faulty or completely fail. The existence of backup nodes, which can substitute the task of centralised nodes if necessary, enhances the robustness against node failures. Only a fully decentralised approach will provide a proper autonomy for the sensor nodes. Last but not least, an ease of use of sensor configuration providing a proper usability of WSNs for non-scientific deployment is still missing. Whereas TinyDB, COUGAR, STOP and TASK are explicitly designed to provide such an ease of use, the other approaches do not even consider this criteria. These approaches still rely on scripting or programming languages and lack of means for customised fault tolerance. While TASK fits best to adaptivity, transparency and convenience, it still needs to be improved in energy efficiency, autonomy and fault tolerance.

However, there exists no approach that associates all introduced criteria. The approach presented here is inspired by some ideas of the discussed approaches and combines these in a new suitable event detection scheme tackling all design criteria. It is quite obvious that fulfilling all criteria up to a level of 100 percent is almost impossible but existing approaches usually tackle only a subset of those. This work introduces a novel concept for sensor network configuration considering all mentioned criteria while being tailored to user needs. It allows to ignore low-level details like node resources, network structures, node availability etc. It further enables the user to work on the description of the event to be monitored including event-related application constraints at a high abstraction level. Specified event descriptions are autonomously configured and monitored, even if it requires to organise collaboration between nodes to deliver the results.

This work introduces the intuitive XML-based ESL that simplifies event definition to a level that is even suitable for non-professionals. It features hardware independent description elements to define complex phenomena and enhances these by tailor-made voting schemes and application constraints. Based on that, a novel, fully decentralised mechanism to autonomously set up distributed event detection, called EDT and a cost efficient means to maintain such an EDT,

are presented. EDTs are efficiently constructed on every device by using a tiny Generating Finite State Machine (GFSM) requiring eight states only. It enables every node to self-divide event queries according to its own resources and self-adapt to the assigned tasks. Simultaneously, the EDT provides the interface for efficient collaboration using a lease-based publish/subscribe approach.

2.6 Introduction to Quality of Information (QoI)

Beside the introduced dependability constraints, the quality of gathered information is crucial for mission- and safety critical surveillance scenarios, since it significantly affects the correctness of the application and the success of the mission respectively. Whereas "Quality of Service (QoS)" is commonly known and widely researched, e.g., in relation to data transport reliability [75], the term "Quality of Information (QoI)" is relatively new and differently defined by the community. The author most agrees to Gillies et al. [22], who define QoI as:

A measure of how well a piece of information delivered to a user described a situation or event of interest.

False alarms are not just a nuisance. In safety-critical applications, e.g., for automatic fire detection or emergency stops in factories, fired events must be well-considered by the decision maker. Wrongfully fired events mean downtime, loss of business and waste the valuable efforts of emergency services. By the way, if alarms sound regularly and unnecessarily, people loose confidence in the system. In the context of this work, it is proposed to determine the significance of events based on behavioural trends of sensor readings. The introduced approach autonomously determines the importance of events for the mission, by considering questions like

How high is the amplitude of measurements?

How strong differ actual readings from previous ones?

How far are respective thresholds exceeded?

Existing approaches of quality-aware sensor networks try to minimise the existing difference between the monitored environment (reality) and the interpreted sensor data [20]. In context of a mission, the goal is to meet a specific information requirement with sufficient quality. Mission requirements have been analysed and bound to sensor capabilities using the {why, when, where, what, who, how} principle [6] or the MARS framework [65]. Beside modelling approaches, probabilistic [22] and quality-aware decision fusion approaches [77, 78], which cope with the correctness of sensor information, have been published. However, all assumptions and approaches require a broad knowledge of application and deployment

constraints. Some of them are highly customised, so that new applications must be applied from scratch. A good example of that dependency is given in [77], where a network of acoustic sensors is used to determine whether a tank is moving through the monitored territory. While this seems to be easy to realise, this approach further claims to distinguish the type of tank with a certain probability, e.g., it determines the chance that the detected tank is of type T80, which belongs to the US army. This represents the measure of QoI here. The downside of this approach is, that unknown types of tanks or other vehicles are either not discovered or identified as noise, because the acoustic profile of such vehicle is unknown to the application. Consequently, a means that is completely independent from the application scenario is in demand. Hence, such means would be applicable to unknown future applications, too. In the best case, this means can be applied automatically to user-defined events of interest. Again, this is considered a necessity for non-scientific deployment.

The main goal is to automatically analyse meta data derived from the event detection process to enrich the final event evaluation result. As already mentioned before, one of the following approaches is used for event detection. In the first approach, all sensors periodically send their readings to a central sink, e.g., a base station, which finally decides about the existence of events. A more efficient way is Boolean style detection, where sensors self-decide about the appearance of events, e.g., based on the exceedance of thresholds. This is presented in Chapters 3 and 4 by introducing the EDT. Whereas the first approach is inefficient in view of energy consumption and traffic, the latter is not accurate enough to allow further decision about the quality of gathered information.

The primary goal of the means presented in Chapter 5 is to upgrade the detection facilities of WSNs applying the event detection concept of EDTs with particular regard to applicability and efficiency. It examines behavioural trends in sensor readings to indicate the significance of actual measurements in relation to the configured event description. Measured data is investigated in detail to finally attach an significance indicator i_S to each event. This automatically generated indicator shall support users or overlaying systems in decision-making. Similar statistical approaches that analyse previously gathered sensor readings such as MASTAQ [26] already exist but these are built on own detection engines.

Chapter 3

Complex Fault Tolerant Event Definition using the Event Specification Language (ESL)

This Chapter presents an architectural overview of the event configuration system and introduces the Event Specification Language (ESL) [47, 46]. The architecture highlights the usage and correlation of ESL and EDT while separating user and machine oriented parts. It further draws a line between programming options of the user and the autonomous configuration part of the system. The major goal of the ESL is to provide means, which allow to define event detection in a straightforward and intuitive way. The ESL is an enabling technology for easy event definition and in-network processing. It provides ease of use for application programming allowing the user to ignore low-level details of the sensor network and to concentrate on a high abstraction level. Namely this is the event itself and its related constraints. By that, it can be used by human users but might as well become part of a "middleware" like approach.

3.1 Architectural overview

Figure 3.1 overviews the architecture of the event configuration system based on EDTs described via the ESL. The system consists of two major components, the event description generator at the user's device and the event configuration environment on every sensor node. Detailed descriptions of the event configuration steps including respective architectural details are provided throughout the next Sections.

At the user's device, the XML-based ESL is used to describe events of interest.

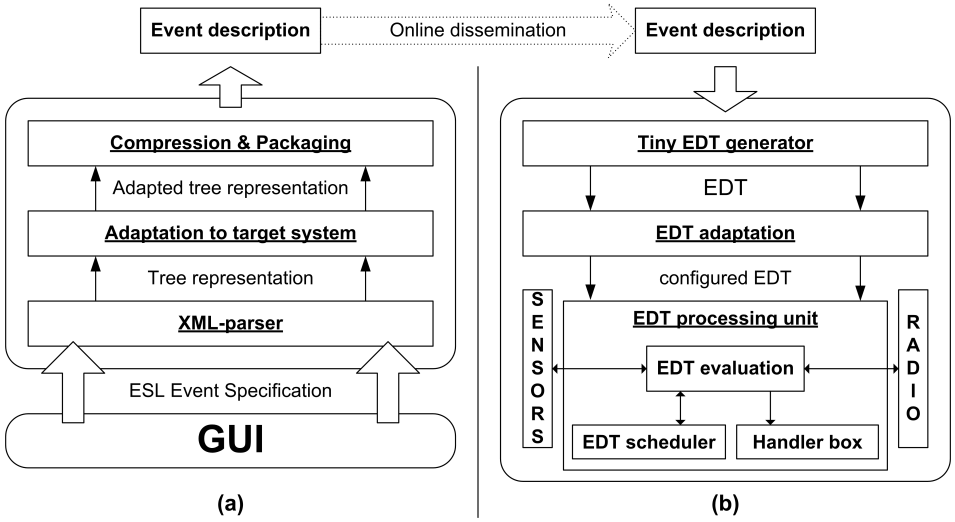


Figure 3.1: Architecture of the event configuration system. It consists of two major components, the event description generator at the user's device (a) and the event configuration environment (b) on every sensor node. Event specifications are disseminated in the network as event descriptions.

Such a description is called event specification. An event specification contains the involved sensing features as well as event related constraints concerning the spatial dimension, detection intervals or the level of fault tolerance. It further specifies associated handlers that are triggered by that event. Here it shall be emphasised that an appropriate event specification is sufficient for successful configuration of the sensor network. Every processing and transformation step based on the event specification and its derivations is automatically done by the event configuration system. It is quite obvious, that general event specifications cannot be uniformly transferred to different sensor nodes. Hence, every event specification passes three steps before being distributed in the sensor network. First, an XML parser generates the respective tree representation. The XML elements of that tree are adapted to the targeted sensor system, i.e., conversion of values for sensing, renaming of identifiers and functions etc. Finally the adapted tree is converted and compressed into a deployable event specification of minimal size, called event description. These event descriptions are distributed in the sensor network for initial event configuration, updates and deletion as well.

On the sensor nodes, the event configuration environment processes every incoming event description to generate the respective event representation as an EDT. According to the sensing features and resources provided by the node, the EDT is split into local and remote parts. Local parts can be evaluated by the node itself, whereas remote parts have to be requested from external sources, e.g., from neighbouring nodes. After further adaptations and configurations of event

related constraints, the final EDT is integrated to the EDT processing unit. The EDT processing unit autonomously collects required sensor readings, frequently evaluates the EDT with respect to the configured detection interval, manages necessary collaboration with other nodes and triggers associated handlers in case of positive event evaluation. Additionally, the EDT processing unit is enabled to administrate and process several EDT at the same time. This is a prerequisite to ensure proper flexibility by allowing the sensor nodes to execute several tasks simultaneously. An integrated updating mechanism enables to replace event specifications analogous to code update means, e.g., as those provided by Contiki [17]. Such a feature allows easy reconfiguration or recalibration of already deployed sensor networks.

3.2 Specifying complex event detection

An event specification associates required sensing capabilities with adaptable requirements for distributed event evaluation by voting and customisable application constraints. The use of the ESL for event specification enables to define precise operating thresholds for heterogeneous sensing capabilities (primitive events) and to combine those by logic operations to specify complex events. Configurable execution intervals and appropriate event handlers can be assigned to events. Further, a region defining the spatial resolution as well as customised voting constraints can optionally be defined for each event. The guidelines for a distributed event evaluation by voting allow to specify the expansion of the voting region or the preferred number of voting participants. Further, exceptional timing criteria can be specified that limit the times available for the voting procedure or the event evaluation time. By that, the event definition itself can already provide fallback solutions for the sensor node to independently circumvent or handle foreseeable faults, e.g., missing votes.

3.2.1 Structure of an event specification

The ESL is a dialect of XML[1] and specifies events within the tag <EVENT>. The complete XML-Schema of the ESL is attached in Appendix A.1. An event specification consists of three mandatory and two optional elements, which are represented by own tags. The <SENSORDATA> element defines the required sensing capabilities for primitive or complex events respectively. Each event specification further contains an <EXECUTION> element that states the frequency of event detection. Appropriate processes, which are triggered upon positive event evaluation, are listed in the <CONSEQUENCE> element.

[1] It is quite obvious that XML is not well suited for use on sensor nodes. Event specifications are pre-parsed by a language interpreter into appropriate event descriptions before deploying them on sensor nodes. The respective parsing process is introduced in Section 3.5.

To improve the event specification complexity, two optional elements enable to fine tune the event observation behaviour. The <DIMENSION> element defines the expansion of the *region of a certain event*. The event region is configured around the node, e.g., as a circle, ball, square, cube or number of hops. It contains all devices, allowed to participate in a distributed event detection. If this element is omitted, the 1-hop neighbourhood is considered the default event region. The <VOTING> element assigns optional constraints for fault tolerant event evaluation by voting, i.e., regions, deadlines, preferred number of votes etc. A voting procedure is not reasonable for every application due to the required overhead. Therefore this element is optional too. Some event detection scenarios may not require voting in favour of less overhead or acceptable loss of accuracy.

For configuring several events simultaneously attributes are embedded in the <EVENT> element. An event-"id" assigns a globally unique identifier to events, which enables to associate requests and updates to a certain event. The "version" number identifies different versions of the same event specification. It reduces maintenance and online reprogramming complexity. Incoming event specifications with higher version numbers substitute all older versions of the targeted event specification, which are deleted to save memory. For removing events from a network, an empty event specification containing the event identifier only is used. The "priority" attribute provides assignment of priority levels to every event to support multi-event evaluation. Consider a sensor network gathering temperature readings for climate control that is used in parallel to detect forest fires. In such a setting the detection of forest fire would have the higher priority because it is a safety-critical event. Currently the language provides three optional priority levels, which are "high", "normal" and "low", whereas "normal" is the default value if not explicitly specified. There are already some Operating Systems (OSs) for sensor nodes such as Realtime Event FLow EXecutive (REFLEX) [70] that provide priority driven task scheduling, which could be applied to implement this language feature directly on the OS level.

To overcome the problems of varying context, fluctuating environment and node mobility sensor networks must frequently self-adapt to the current situation. Especially if nodes collaborate with each other, these relations may suddenly be disturbed or not available any longer. Hence, sensor nodes must provide means to renew collaboration since prevention of such changes is hard or even impossible. This again requires a processing and communication overhead. To allow the programmer to customise the adaptiveness and efficiency of the communication scheme used for collaboration between neighbouring nodes, the "lease" parameter and the "reliableMode" attribute are specified. The "lease" defines the frequency of adapting collaborative relations between neighbouring nodes. In other words, it specifies the time lag between two adaptation phases where usual event processing takes place. Short lease intervals (small lease factor) pro-

```
<EVENT id="example" version="1" priority="high"
       lease="6" reliableMode="yes">
  <SENSORDATA>   ... </SENSORDATA>
  <EXECUTION>    ... </EXECUTION>
  <CONSEQUENCE>  ... </CONSEQUENCE>
  <DIMENSION>    ... </DIMENSION>
  <VOTING>       ... </VOTING>
</EVENT>
```

Listing 3.1: Basic structure of an event specification. It displays version "1" of the event "example", which assigns a "high" priority and a lease factor of "6". Further the "reliableMode" is enabled.

vide a high adaptation rate whereas long lease intervals can significantly reduce the number of messages and the energy consumption. For example, highly fluctuating WSNs should apply short lease factors to cope with changing topology and moving nodes. In contrast to that, WSNs with static deployment may use long lease factors to reduce number of required collaboration messages. Further description of this parameter is given in Section 4.4.2.

The "reliableMode" attribute permits to choose between a higher reliability in data exchange or a reduced energy consumption. Enabling the "reliableMode" instructs to explicitly acknowledge every data exchange that is associated to a certain event. The reliable mode consequently requires a communication overhead but enhances the reliability of detection. Thus, safety-critical events should make use of the reliable mode, whereas simple data collecting scenarios could omit the required overhead in favour of less energy consumption. It is quite obvious that configuration of both parameters strongly depends on the application as well as the application context. Listing 3.1 shows the basic structure of an event specification containing the mentioned elements. This specification exemplary lists all possible elements. The elements configuring the applied sensing features, the evaluation interval and respective event handlers are mandatory. The elements assigning the event region and the voting guidelines, which are displayed in gray, are optional. Detailed descriptions and configurations of these elements are presented in the next Sections.

3.2.2 Definition of complex phenomena

Appropriate combinations of heterogeneous sensor features enable more precise and complex event detection facilities [62]. Almost all sensor network applications define threshold values for certain measurements, called primitive events, and fire an event if current sensor readings match or exceed these values. The ESL provides means to easily combine several sensing capabilities and respective primitive events to complex ones within the <SENSORDATA> element.

Primitive events can be defined as the sensor reading matching an exact value or through single-bounded intervals the sensor reading falls into by using the following relational elements:

- <EQUAL>,
- <LESS>,
- <GREATER>,
- <LESSOREQUAL>,
- <GREATEROREQUAL>.

These elements define respective binary relations between two elements, which are variables, constants and/or mathematical functions defined on top of these. The result of a relation is of Boolean type. The usage of these elements is discussed in the next Section. Please note, except for the equality relation these elements are not commutative and hence, require correct order of the assigned elements. It is quite obvious that ($temperature < 10$) is semantically different from ($10 < temperature$), for example. To support definition of complex phenomena, configured thresholds can be composed by logic operations. Logic operations are specified by own tags, namely:

- <AND>,
- <OR>,
- <NOT>.

Similar to the relational elements, the elements <AND> and <OR> define a respective logic operation between two relational or two logic elements or a mix of both. Since these operations are commutative, the order of the linked elements is irrelevant. The <NOT> element specifies an unary operation and can only be used on top of one relational or logic element. As known from Boolean algebra, it inverses the Boolean result of the underlying element. Of course, it is also possible to link several logic operations together. Logic elements further allow to define 2-bounded intervals for certain sensing capability by combining several primitive events. For example, measuring a temperature between 20 and 25 results in a combination of two primitive events, i.e., ($temp > 20$) AND ($temp < 25$). A respective <SENSORDATA> is exemplarily given in Listing 3.2. This event configuration evaluates to TRUE if temperature readings are between 20 and 25 centigrade.

To simplify matters only three logic elements are available, but these are quite enough to define every possible logic combination. Supporting more language elements may slightly increase the usability for end-users but even implies to implement more complex interpretation means on the sensor nodes. This would require to use more processing and memory resources. For this reason and to keep the language quite simple, the integration of further logic elements like NAND or NOR, is omitted. This is vitally important for implementing a language interpreter on sensor nodes, which usually provide scarce resources only.

```
<SENSORDATA>
  <AND>
    <GREATER>
      <VARIABLE> temperature </VARIABLE>
      <CONSTANT unit="centigrade"> 20 </CONSTANT>
    </GREATER>
    <LESS>
      <VARIABLE> temperature </VARIABLE>
      <CONSTANT unit="centigrade"> 25 </CONSTANT>
    </LESS>
  </AND>
</SENSORDATA>
```

Listing 3.2: Example sensor data element of a composite event, which detects temperature between 20 and 25 centigrade.

Mapping of sensing capabilities, thresholds and mathematical functions

Configuring events primarily requires to set thresholds for certain sensor readings using relational elements. As mentioned above, relational elements enable to define respective relations between two variables, constants and/or mathematical functions. A variable identifies a sensing capability and is defined by the <VARIABLE> element. Thus, the value of a variable is given at run-time by sensor readings. In contrast to that, the <CONSTANT> element defines a constant value, which can be used as a threshold. Constants usually require to set an additional measuring unit. The "unit" attribute allows to assign different units to constants, e.g., time and distance units like "seconds", "minutes", "meters" etc. In certain cases constants may not require a unit, for example if a pure quantity is in demand. Such a constant is specified without the "unit" attribute. Conversion of the specified constants with respect to the hard- and software used on the sensor nodes, e.g., converting seconds to milliseconds if necessary, is task of the language interpreter and is not of concern for the user. That allows for straightforward event definition, even practicable for the non-professional user.

Whereas relational and logic elements generate Boolean results, variables and constants apply numerical values. To support a broad usability as well as to enable conversion of values, the ESL enables to define binary mathematical functions using variables, constants or results of further functions as parameters. Consequently, functions result in numerical values. The ESL provides the following functions:

- <SUM>,
- <DIFFERENCE>,
- <PRODUCT>,
- <QUOTIENT>,
- <MODULO>.

Please take note, differences, quotients or modulo functions are not commutative. Just as mentioned at relational elements, even here the order of the applied parameters is crucial.

By design, the ESL also allows to define relations between two sensor readings or intermediate results of equal type. Since this is feasible for variables and functions, e.g., for comparing inside and outside temperature readings of a building, its application is rather useless for two constants. The result of such relation is of constant Boolean type, too. In that case, the event specification is redesigned and the Boolean result of such relation is directly inserted instead.

3.2.3 Addition of execution constraints and associated handlers

Energy consumption is an essential and very critical issue when designing WSN-based applications. Therefore the ESL provides means that help to adjust the energy consumption of the event evaluation process. Sensor nodes provide different modes of operation that result in significantly different amounts of energy consumption. Active modes like data processing or data transmission are draining the energy resources much more than passive modes such as sleeping [55]. Thus, active periods must be kept as short as possible to reduce energy consumption to a very minimum. On the other hand, extensive passive periods may reduce the accuracy and reliability of event detection. Real-world phenomena are usually subject to different temporal resolution, which must be considered for event specification as well. For example, the acoustic wave of an explosion can only be detected within a few milliseconds and hence, requires a short sensing interval.

When a node may switch to a power saving mode depends highly on the application running. Thus, an event specification contains an <EXECUTION> element to configure application-oriented execution constraints for each event. The ESL supports various event constraints that provide options for customisation of execution, dimension and voting constraints. In general, event constraints are defined by the pattern given in Listing 3.3. An event constraint is defined by its name and sets a “relation” (by attribute) with a constant. The only exception is the “InBetween” relation, which requires two constants as lower and upper bounds defining the valid range of values. The <TIMEINTERVAL> element defines the event evaluation frequency as time interval. Time intervals can be quantified by acceptable periods or exact time slots that must be adhered to.

The <CONSEQUENCE> element is the last mandatory component of an event specification. It links procedures to an event. Theses procedures, called event handlers, have to be executed in case of positive event evaluation. Every event handler is listed as a <TRIGGERHANDLER> element, which contains the name of the event handler. Specifying several event handlers for a single event is allowed and all of them are executed in the sequence listed, if that event occurs. Since event handlers trigger available functions or processes at the sensor

```
<$NAME$ relation="EqualTo"
                 "LessThan"
                 "GreaterThan"
                 "LessOrEqualTo"
                 "GreaterOrEqualTo"
                 "InBetween">
  <CONSTANT unit="milliseconds"> 100 </CONSTANT>
  <CONSTANT unit="milliseconds"> 250 </CONSTANT>
</$NAME$>
```

Listing 3.3: Pattern of an event constraint. An event constraint is defined by its name ($NAME$), a "relation" attribute and a constant as threshold. The only exception setting the "InBetween" relation as the relation attribute since it requires to define the lower and upper bound by two constants.

nodes, those must be adapted to the target platform and the respective OS by the language interpreter as well. For example, a general event handler such as "sendalert" could be mapped to a respective interrupt routine of the OS.

3.2.4 Determining the region of event

Besides the temporal resolution, which is configured in the execution element, also the spatial expansion of every event depends on the phenomenon to be sensed. Wireless sensor nodes can communicate up to 1000 feet (approx. 300 meters) [23] but many phenomena usually appear only locally. For example, in an environmental surveillance scenario temperature changes usually appear widely, whereas the size of an emerging fire is relatively small but has to be detected as well. Hence, the ESL allows to describe the expected spatial expansion of the phenomenon to be sensed as *region of event*. That especially is of interest if sensor nodes jointly share their resources for collaborative event detection. For reliable event detection, collaborating nodes must know whether they share a certain region of event. The ESL configures this valid *region of a certain event* within the <DIMENSION> element. The *region of event* can be specified by one of the following event constraints:

- <CIRCLE>,
- <SQUARE>,
- <BALL>,
- <CUBE>,
- <HOPS>.

According to their names, these elements enable to define 2-dimensional event regions, i.e., <CIRCLE> and <SQUARE>, as well as 3-dimensional ones, i.e., <BALL> and <CUBE>. Rather dedicated to WSN is the <HOPS> element defining the number of hops as valid event region. If the <DIMENSION> element is omitted, the 1-hop neighbourhood is taken as default event region, which is

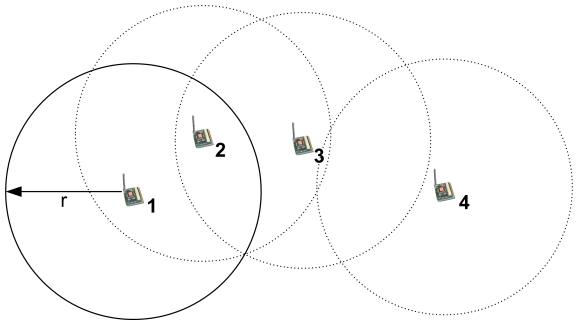


Figure 3.2: Example deployment of nodes with circle event regions configured by radius *r*. Whereas node *4* is isolated, node *1* shares its event region with node *2*, node *2* may collaborate with *1* and *3* and *3* may evaluate events with node *2*.

determined by sending range. These virtual event regions are spanned around each sensor node. In other words, each sensor node is the centre of an event region and can be part of other event regions spanned by neighbouring nodes as well, see Figure 3.2.

The specification of listed elements follows the pattern of event constraints, as introduced at Listing 3.3. The assigned constants here define the radius of circles and balls, the length of the edges of squares or cubes, as well as the number of allowed hops. Wireless communication transmits in 3-dimensional spatial resolution and therefore the definition of 3-dimensional event regions is needed. The intention of additionally providing 2-dimensional regions is to decrease the configuration complexity of events, e.g., for simplifying monitoring applications for floors lying upon each other in office buildings, where only sensors located at the same floor are allowed to collaborate.

To provide locally restricted events, the sensor nodes must be either enabled to determine distances between relevant neighbours or to limit the sending range with respect to the specified region of event. Except for using the number of hops as a region, there is a necessity to have node's positions available to determine the nodes sharing a certain region. That position data can be given at deployment or be retrieved during runtime by methods such as Global Positioning System (GPS) [10], triangulation using Received Signal Strength Indication (RSSI) or Chirp Spread Spectrum (CSS) [3, 44], or transmission power [7, 8]. Reduction of the sending range by limiting transmission power with regard to the specified event region seems to be the most efficient implementation, since it implies a reduced power consumption, too. However, it may be difficult or even impossible to realise this method on various sensor hardware and most important, it is not stable enough for reliable application in real deployments. It is well-known, that the propagation of radiowaves varies continuously and suddenly as well due to changing environmental conditions and context. It is unreasonable to believe that

adjustable transmission power can reliably cope with those conditions as being expected in ubiquitous applications.

3.2.5 Customising voting preconditions

Due to the fault probability in sensor networks, fault tolerant event detection is absolutely essential for ensuring reliability. Exploitation of redundant data sources is well known to offer an enhanced reliability of application. Voting mechanisms have proven to be functional to provide such feature in WSN, too. Since different applications demand different fault tolerant behaviour, the preconditions of event evaluation by voting can be customised for every event. The optional <VOTING> element enables to list voting constraints for definition of the voting region, the preferred number of voting devices and time limits for the voting process.

Analogue to the <DIMENSION> element, the valid region for voting can be defined as ball, cube, the number of hops etc. The voting region contains all devices allowed to participate in the voting. Of course, this is restricted to devices that can evaluate the same event. The voting region may differ from the event region, e.g., the voting region can be smaller to locally obtain better voting results. Since voting evaluates the local detection results of a sensor node, it may be suitable to include nodes in this process only, which are very close to the initiating node. In contrast to that, the event region follows from the expected spatial expansions of the phenomenon to be sensed and identifies all nodes allowed to collaborate. If no explicit voting region is defined, the event region is taken as default, which is at least the 1-hop neighbourhood.

Besides the voting region, exceptional voting conditions must be considered as well. An insufficient number of available voters may exist if the chosen event region is too small or the density of devices is not high enough. Furthermore, the detection of events may require to fulfil certain timing constraints. Delays in the voting process may especially reduce the reliability of mission- or safety-critical applications. The following elements define event constraints that allow to skip or abandon the voting process and still trigger configured processes if necessary. The <NUMBEROFVOTES> element adds individual voting objectives. It allows the user to adjust the preferred number of voting devices within the voting region, which can be fixed or limited by a minimum or a maximum threshold. Hence, the voting finishes if the initiating node has received the sufficient number of votes as specified in this element. The <DEADLINE> element specifies the period of time after which the event evaluation and necessary voting procedures need to be finished. That enables the user to introduce time criteria as it may be necessary to meet provide a sufficient response behaviour, e.g., for safety-critical applications such as emergency stop. If the deadline is reached the current voting status is used as the final result of evaluation.

The <NODEVICES> element considers an exceptional state, when no other device is a priori available within the voting region and hence, necessary event handlers are triggered without voting to save energy resources. That case is currently detected by failed voting requests but could also be identified by analysing routing information if accessible. Since operational conditions may change over time, this case must be verified periodically. A voting fails if no other devices answer the voting request. In that case, the node omits voting for a number of intervals. The number of these intervals is determined by the constant given as attribute of the <NODEVICES> element. After theses intervals the node triggers the voting as usual. To enable multi-purpose exception handling several voting constraints can be listed simultaneously. If any listed constraint is satisfied, the voting procedure terminates with the actual voting result. A proper combination of these constraints enables the user to fine-tune the voting behaviour and determine stop criteria regarding deployment and application conditions. In the majority of already published approaches these parameters are fixed.

3.3 Reactive Majority Voting (RMV)

Voting enhances the reliability of detection but in turn requires an overhead in time and energy consumption. Voting might be triggered in case of local positive event evaluation or if a measured value deviates more than a certain percentage from previous measurements. The latter case is presented and discussed in Chapter 5. Here, the voting in case of local events is of interest. All nodes within the voting region are authorised to participate in the voting and may submit their own event detection result to the initiating node. The initiating node collects all votes and counts the positive as well as the negative ones. Finally this node decides based on the majority of the votes whether or not an event was detected. The initiating node may trigger further actions if necessary. The presented algorithm also interprets a tie situation in voting as a positive result, since it shall rather detect a non-occurred event than to miss one. This is considered as necessary issue especially for mission- and safety-critical applications.

To reduce the energy consumption, Reactive Majority Voting (RMV) is introduced where voting is done if and only if it is necessary. A reactive voting algorithm provides a high efficiency while achieving the same or similar reliability as other approaches, even in case of faults. Existing approaches make use of fixed voting regions as well as selected nodes for decision making, which collect and evaluate all measurements in their voting region. Fixed voting regions are usually built in the initial phase or are given at deployment and hence, are not flexible enough to reliably detect different phenomena of varying spatial resolution. Further, fixed voting regions cannot suitably cope with changes in the environment and varying node density and even less with mobile nodes. In

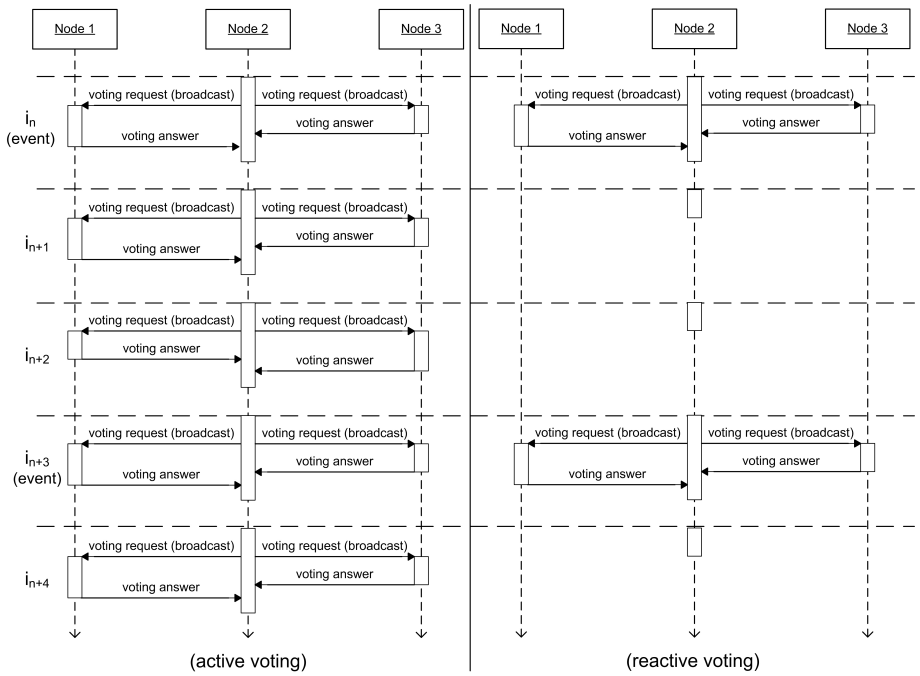


Figure 3.3: Comparison of applying active MV and RMV on a sequence of event detection intervals. Active MV requires to perform voting at each detection interval, even if there exists no noticeable phenomenon (event). In contrast to that, RMV needs to perform a voting on event only and hence, significantly reduces the number of transmission and voting procedures. According to this, RMV provides a high energy efficiency.

existing approaches only the decision making node finally is able to identifies the phenomena and hence, these approaches require to perform a voting at each detection interval even if no event is indicated by current sensor readings.

In contrast to that, establishing unfixed local voting regions around the nodes enables each node to independently trigger a voting on demand. On detecting an event, RMV requests all nodes in the assigned voting region to perform an usual MV. In the case of negative local event evaluation results, RMV allows to abandon voting and switch to the sleeping mode immediately. That significantly reduces the energy consumption. This is underlined by Figure 3.3, which compares the usual (active) MV to the proposed RMV. It displays application of both methods in a sequence of event detection intervals considering the cases of non-event and event (i_n and i_{n+3}), which clearly shows the benefit of RMV. To give a proof of concept, the possible evaluation results are analysed in the following. It is discussed how RMV detects and overcomes failures efficiently. Local event evaluation may result in four possible states, depending on the actual existence of the phenomenon to be measured. These are:

Correct positives are events identifying an existing phenomenon. In that case, voting cannot be avoided to distinguish these from false positives but should result in positive voting evaluation as well.

False positives are erroneous reported events of non-existing phenomena. Wrongfully detected events are usually identified by voting and are overruled by other devices.

Correct negatives rightly identify no noticeable event. Hence, also other devices usually do not identify an event. Thus, voting becomes useless since it would result in a negative voting result as well.

False negatives are wrongly non-identified events caused by faults in sensor nodes or sensing devices. These events are most likely detected by other devices in the event region. Hence, these devices trigger the necessary voting, which should identify faulty state. A special case of this is the failure of a voting initiator while the voting is still in progress. If the event actually exists, the other nodes participating in the voting will fire the event after their next detection interval. This introduces a delay of only one detection interval.

To conclude, RMV offers the same or similar reliability but gets by with significantly less voting procedures. In fact, it reduces the number of voting processes by a factor of $1/p_t$, whereas p_t is the event probability. For surveillance and event-based applications it is considered sufficient to evaluate only detected events. This allows to reject *False positives* and to identify the nodes that most probably correctly detected an event. In addition, the ESL enables the application engineer to fine-tune voting constraints or even to completely switch off the voting for every event. That allows for more economic power consumption than the majority of the already published ideas, in which voting is enabled by default. Customised majority voting provides more precise and flexible detection of events than voting within fixed regions, but could even be improved by customising the voting algorithm. Hence, the ESL may be extended to provide other voting algorithms like TIBFIT [34] or CWV [64] as well.

RMV does not rely on a certain failure model. The focus is on the correctness of measured events as well as to reduce the energy consumption. There already exist voting algorithms that cope with stucked measurements and Byzantine faults [14, 27, 41]. Those require an enormous overhead in processing and communication by at least a factor of two or three compared to usual MV, which is yet inefficient compared to RMV. Further work intends to integrate selection means for the voting algorithm that allows the user to choose the one that best meets the application requirements regarding performance and communication overhead.

3.4 Fire detection scenario - An illustrative example

To illustrate the ESL, this Section introduces fire fighting systems in homes as an example scenario. Fire detection is a well descriptive vehicle to underline the mission- and safety-critical aspects in event detection. Of course, reliable detection of fires is of utmost importance because undetected fires pose a threat to life of people in the building.

Besides other criteria, a fire can be detected by monitoring the ambient temperature, the emission of smoke or the existence of carbon monoxide. Traditional and widely used fire detectors set off a fire alarm if monitored smoke or carbon monoxide emissions exceed a given threshold. Also changes in temperature can be analysed to indicate or even detect a fire [25]. In spite of using well-engineered sensing devices these methods are still vulnerable to false alarms, e.g., triggered by smoking, burnt food or influences of various heat sources. Each detection method is suitable to detect fires indeed, but proper fusion of all systems enhances the reliability of detection and decreases the false alarm probability at the same time. It further enables to detect different kinds of fire that also reveal different physical properties [4], e.g., flaming and smouldering fires. Thus, a proper fire fighting system should apply temperature, smoke and carbon monoxide detectors simultaneously.

Listing 3.4 displays an event specification that can be used in fire detection scenarios. To decide about the existence of fire, each detection method usually determines a fixed threshold. This example proposes to apply 100 ppm (parts per million) as threshold for carbon monoxide, 1.1 percent as smoke limit, and 353 Kelvin (80 centigrade) as ambient temperature limit. Whereas the existence of carbon monoxide is a good stand-alone indicator for fire [4], temperature and smoke readings should be suitably combined to gain a higher false alarm safety. Therefore, smoke and temperature thresholds are linked using a logical AND and combined with the carbon monoxide threshold using a logical OR. Hence, an event *fire* is detected if either the carbon monoxide readings exceed 100 ppm OR both smoke AND temperature readings exceed their assigned thresholds. In case of having evaluated the *fire* event to be positive, the sensor node triggers the "sendalert" event handler. A radius of 2.5 meter around the sensor nodes is assumed a suitable region for distributed detection. Hence, the dimension element defines that region as a ball specifying a maximum radius of 2.5 meters. Please note, the dimension element is considered only for distributed collaboration in event detection between sensor nodes. The necessity of defining the region of the *fire* event will be presented together with the collaboration algorithm in the next Chapter, see Section 4.4.2.

In addition, three constraints are determined for final event evaluation by voting. The voting process demands at least three voting devices for evaluation. Hence, the voting finishes either if at least two additional devices have sent their

```
<EVENT id="fire" version="1" priority="high" lease="6" reliableMode="yes">
  <SENSORDATA>
    <OR>
      <GREATEROREQUAL>
          <VARIABLE> carbon monoxide </VARIABLE>
          <CONSTANT unit="partsPerMillion"> 100 </CONSTANT>
      </GREATEROREQUAL>
      <AND>
        <GREATER>
          <VARIABLE> temperature </VARIABLE>
          <CONSTANT unit="Kelvin"> 353 </CONSTANT>
        </GREATER>
        <GREATEROREQUAL>
          <VARIABLE> smoke </VARIABLE>
          <CONSTANT unit="percentage"> 1.1 </CONSTANT>
        </GREATEROREQUAL>
      </AND>
    </OR>
  </SENSORDATA>
  <CONSEQUENCE>
    <TRIGGERHANDLER> sendalert </TRIGGERHANDLER>
  </CONSEQUENCE>
  <EXECUTION>
    <TIMEINTERVAL relation="EqualTo">
      <CONSTANT unit="seconds"> 10 </CONSTANT>
    </TIMEINTERVAL>
  </EXECUTION>
  <DIMENSION>
    <BALL relation="LessOrEqualTo">
      <CONSTANT unit="meters"> 2.5 </CONSTANT>
    </BALL>
  </DIMENSION>
  <VOTING>
    <NUMBEROFVOTES relation="GreaterOrEqualTo">
      <CONSTANT> 3 </CONSTANT>
    </NUMBEROFVOTES>
    <DEADLINE relation="EqualTo">
      <CONSTANT unit="seconds"> 3 </CONSTANT>
    </DEADLINE>
    <NODEVICES relation="EqualTo">
      <CONSTANT> 6 </CONSTANT>
    </NODEVICES>
  </VOTING>
</EVENT>
```

Listing 3.4: Example of an event specification for fire detection scenarios.

own results or if another exceptional condition is met. The voting procedure is cancelled by exception if no other devices are a priori available in that event region or if the initiating sensor node receives no results from other devices within three seconds. In case the voting cannot be accomplished successfully, the initiating sensor node triggers the “sendalert” handler latest three seconds after itself evaluated the event to be positive. Since no explicit voting region is specified, the event region is applied as the voting area.

3.5 Generation of deployable event descriptions

XML-styled event specifications provide flexible and easy-to-use configuration means for widely used sensor networks beyond the scope of research, even for non-experts in the field such as medical employees adapting them for customised patient monitoring. Nevertheless, XML is oversized for direct use on WSNs, which are subject to strict energy and memory constraints. To minimise the calculation effort on the sensor nodes as well as to minimise the amount of data to be transferred, event specifications are pre-parsed to generate smaller versions before in-network deployment, called event descriptions. These event descriptions are applied for initial event configuration as well as for event updates, i.e., event reconfiguration or deletion. This Section presents implementation details and the workflow of the description generator, which is depicted in Figure 3.4.

The description generator creates hardware-specific event descriptions of universally valid event specifications. The user's only task is to define the event specification. Most suitably this is done via a Graphical User Interface (GUI) as it is displayed in the architecture. Note, GUI is not obligatory to present the content of this work. For demonstration issues the description generator parses text files containing the event specification instead of using a GUI. It is quite obvious, that general event descriptions cannot be uniformly transferred to every sensor platform due to different hardware and software properties [58]. To overcome these problems introduced by the ubiquitousness, the description generator adapts variables, thresholds, handlers, event constraints etc. to the target sensor platform regarding expected hardware, sensing capabilities and the OS. Afterwards, the elements in the adapted event specification are substituted by symbols and compressed into the final minimised event description layout, which can then be used for configuration. Except for creating the event specification, all mentioned steps are fully-transparent to the user and automatically done by the description generator. That allows to keep the event definition process quite simple and intuitive.

The complete ESL description generator is written in Java to enable execution on different devices and platforms providing a Java Virtual Machine (JVM). In particular, the focus is on mobile devices like a laptop or a smartphone. This provides the necessary mobility for envisioned applications in pervasive technologies. In addition, the ESL derived from XML provides ideal requirements to be implemented and used by Web services, too. This offers great potentials to realise remote configuration concepts for ubiquitous technology via the Internet. By class design, the description generator provides classes for each ESL element and an interface for hardware abstraction while regarding a sufficient extensibility of all of them. The corresponding class diagram is given in Figure A.2. The modular structure of the classes related to ESL elements provides interfaces and abstract basic classes (italic class names) for every group of elements, which al-

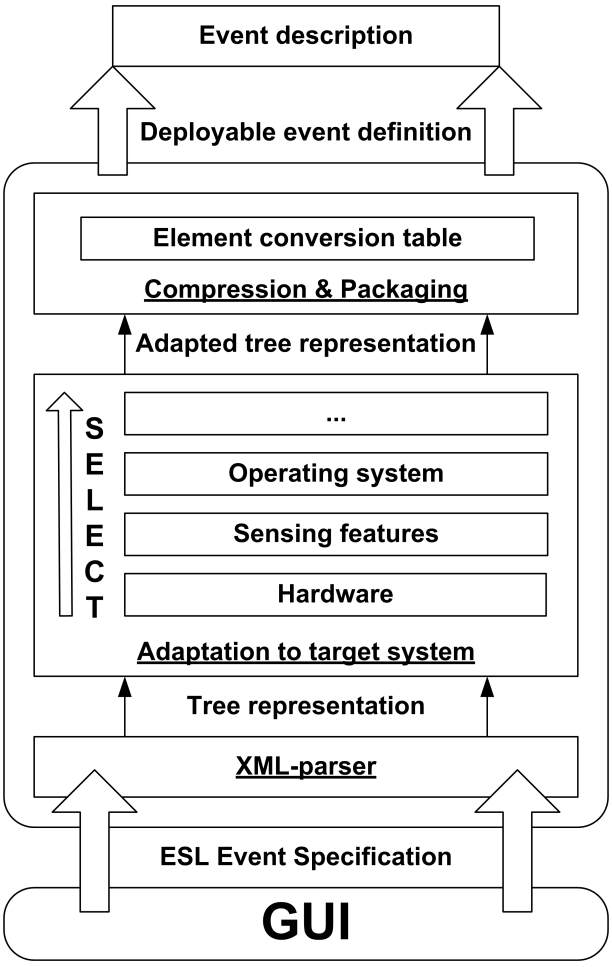


Figure 3.4: Architecture of the ESL description generator.

lows to easily implement future extensions of the ESL. The hardware abstraction layer interface enables to implement the mentioned adaptation to meet the requirements of the target sensor platform. More precisely, this interface provides a couple of functions for harmonisation of general ESL elements. These functions, which are presented in more detail in the next Section, process the elements that need to be adapted and return the customised variants. Hence, each sensor platform only needs to provide a suitable implementation of this interface to gain compatibility to the introduced event configuration concept. That allows to easily create different hardware-specific event descriptions for deployment on various platform. To simplify matters, the following Sections present the generation of the event description for the introduced example scenario while taking the simulator OMNeT++ with an upgrade for WSNs as the target platform. Please note, the generation of event descriptions for other platforms consequently follows the equal scheme but may of course require other customisation.

3.5.1 Adaptation to target sensor platform

Event specifications are first parsed into an in-memory representation of the XML-tree. The following describes the customisation process for meeting the requirements of the target sensor platform. Therefore, the implementation of the hardware abstraction layer interface is used. First, constants and variables are adjusted. The name of variables, which identify sensing capabilities, are modified to internal identifiers as being used on the sensor nodes. In the example scenario, the variables *carbon monoxide*, *temperature* and *smoke* are changed into CO, T and S respectively. Besides different identifiers, assigned constants (thresholds) are converted into those matching the target platform, e.g., equal sensing capabilities may be measured by different sensors with varying physical units. For example, temperature values given in Kelvin are converted to centigrade or time data is converted from minutes to milliseconds if necessary. In the given scenario, the thresholds for temperature and smoke reading need to be aligned. Since the temperature readings in the simulations are measured in centigrade, the threshold of 353 Kelvin is converted to 80 centigrade. Similar to that, the threshold for smoke detection is changed from 1.1 percent to 11 per mille. As another important gain of this processing, it allows to omit the unit of a constant for deployment, since this value has already been scaled to the correct one and can be directly used as the threshold. Hence, it reduces the size of event packages and consequently saves energy required for transmission.

Certainly, the constants used for event constraints must be adapted in the same way as well. Here, the timers of the target platform demand to provide time data in milliseconds. Consequently, the time interval of 10 seconds is replaced by 10000 milliseconds and the deadline is set to 3000 milliseconds instead of three seconds. Finally the event handlers must be adapted to the OS. The name of the event handler is either changed to a function available at the OS or to the number or identifier of an interrupt routine that has to be called on positive event evaluation. These adaptations are suitable to fulfil the requirements of the applied simulator indeed, but most likely it is not enough to remain fully compatible to the bulk of available sensor platforms. A good overview of means to adaptation software to different hardware, especially with respect to resource constraint devices or embedded systems, can be found in [5]. Detailed customisation requirements are expected to emerge from experiences with real deployments in future work.

3.5.2 Creation of event descriptions

After adaptation, all elements are successively transformed into minimised descriptions before being deployed as an event description. An event description consecutively lists all five basic elements of the respective event specification.

ESL-element	Description element
<AND>	&
<OR>	|
<NOT>	!
<EQUAL>	=
<GREATER>	>
<LESS>	<
<GREATEROREQUAL>	>=
<LESSOREQUAL>	<=
<SUM>	+
<DIFFERENCE>	#
<PRODUCT>	*
<QUOTIENT>	/
<MODULO>	%
<TIMEINTERVAL>	I
<NUMBEROFVOTES>	V
<DEADLINE>	D
<NODEVICES>	N
<CIRCLE>	C
<SQUARE>	S
<BALL>	B
<CUBE>	K
<HOPS>	H

Table 3.1: Conversion table containing event specification elements and respective event description elements. Elements of the event specification that are not listed here need not be converted since these are represented by the fixed structure of the event description.

Keeping a given order allows to describe these elements by their content only. More precisely, each description lists the event header, followed by the sensor data element, the event handlers, the execution constraint, the dimension constraint and finally the voting constraints in exactly that order.

In the shortened form, all parameters of the event header are associated to one string. Whereas *lease* and *version* numbers as well as the event *id* are directly taken, the attributes *priority* and *reliableMode* are represented by their first character only. For the given example, the short event header *fire.1h6y* represents version *1* of the event *fire*, which assigns a high priority and a lease factor of *6* while enabling the reliable mode (*y*).

The sensor data element is converted to a minimised prefix (or polish) notation of the respective XML subtree. The prefix notation places operators to the left of their operands. Since the arity of the ESL operators is fixed, which is here one for the NOT and two for all other elements, the result is a syntax without brackets

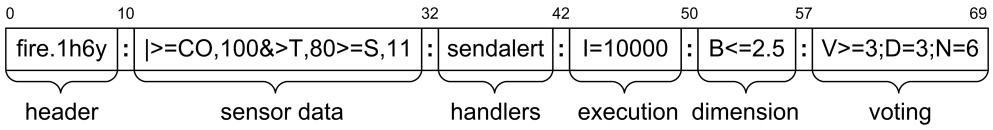


Figure 3.5: Final event description of the introduced fire detection example. This description contains all necessary information for configuring sensing devices according to the event specification. The numbers displayed on top of the description represent the respective offset addresses in the byte stream.

that can still be parsed without ambiguity. Consecutively listed variables and constants can in principle not be distinguished. Variable names are allowed to contain numbers. Hence, an implicit delimitation from the following constant (numbers only) cannot be achieved. Therefore, a constant that follows a variable is additionally separated by a comma. In contrast to that, the case of a variable following a constant is implicitly identified and allows to omit the comma, since variables have to begin with an alphabetic character while constants contain numbers only. To minimise the final description size, the tags of the ESL elements are represented by short symbols, see Table 3.1. These symbols are assigned via the hardware abstraction layer and may differ depending on the target platform. Additionally please note, usually the - or the Δ symbolise the mathematical difference operation. Herein, the # is used to symbolise the difference operation. Automatic distinction of the - symbol when being used for signed constants and differences as well, is unsuitable since it increases the parsing complexity. The Δ symbol is not used because it is not contained in the standard ASCII character set and may be not supported by each sensor platform. For the same reason, the use of symbols $\leq$ and $\geq$ is deprecated. The character sequences <= and >= are used to describe the respective operations.

As the next part, the event handlers are added to the description, which are consecutively listed and separated by comma. Finally, the event constraints are minimised. An event constraint consists of its short identifier, followed by the symbolised relation attribute and one or two constants respectively. The event description is completed by adding the remaining elements containing constraints only, these are the execution element, the dimension element and the voting element. The voting element consists of at most four constraints, i.e., the voting region, the number of votes, the deadline and the no-devices element, theses elements are additionally separated by semicolons. Since event constraints have a fixed structure these are not parsed by the introduced EDT-engine to keep the GFSM simple. Event constraints are described as infix notation and are evaluated by string matching operations according to the regular expression given in (3.1).

$$[A-Za-z]^{+}[< \mid > \mid = \mid <= \mid >= \mid <>]^{\{1\}}[0-9]^{+}([,]^{\{1\}}[0-9]^{+})^{\{0,1\}} \quad (3.1)$$

Event descriptions are transmitted as Byte-Streams to the sensor nodes whereby the basic elements are separated by colons. Unspecified elements, or omitted optional ones respectively, are represented by an empty placeholder. Hence, an omitted basic element result in two consecutively listed colons. Ensuring such strict layout reduces the size of event descriptions compared to the XML-styled event specification by an average factor of ten. For instance, that allows to scale the size of the introduced event specification for fire detection by a factor of 13 from 949 Bytes plus white-spaces down to 69 Bytes provided as event description. Figure 3.5 displays the respective event description.

Chapter 4

Deployment on Sensor Nodes as Event Decision Tree (EDT)

This Chapter presents how sensor nodes are configured according to ESL-designed event specifications. More precisely, it describes the conversion of event descriptions into their processable form as Event Decision Tree (EDT) [48, 49]. It provides details about configuration and maintenance of those, which are performed by the EDT-engine on each sensor node. This includes implementation features of the generation and evaluation of the EDT as well as dividing complex events into less complex ones based on the sensing facilities of individual sensor nodes. It further introduces efficient means to detect nodes for collaboration, which can provide missing information to evaluate the complete EDT. Finally, simulation results of a prototype implementation applied to various failure scenarios underline the cost-efficiency of the presented approaches.

4.1 Architecture of the EDT-engine

On the sensor nodes, the EDT-engine configures the sensor node with respect to each received event description. The architecture of the EDT-engine is displayed in Figure 4.1. First, the *EDT generator* processes the sensor data element of every incoming event description in a tiny GFSM with eight states. As a result it generates the respective event representation (phenomenon to be sensed) as an EDT. Depending on the sensing features and resources provided by the node, the *EDT adaptation* splits this EDT into local and remote parts. Local parts can be evaluated by the node itself, whereas remote parts have to be requested from external sources, e.g., from neighbouring nodes. The *EDT adaptation* further adapts and configures event related constraints as parameters of the EDT, i.e.,

event regions, handlers, voting constraints etc.

The final EDT is integrated to the *EDT processing unit* that maintains the compliance with all parameters of the configured EDTs. The *EDT processing unit* consists of the *EDT evaluation*, an *EDT scheduler* and a *Handler box*. The *EDT scheduler* autonomously schedules all EDTs with respect to their configured evaluation intervals. This schedule is currently implemented by timers assigned to each EDT. On timer wakeup, the respective EDT is enqueued into a queue that holds all EDT pending for execution. That guaranties the evaluation of all EDT, even if several of them are triggered simultaneously or with short lags. This queue is in principle a First In First Out (FIFO) queue, but as a second criteria for scheduling the enqueued EDTs are sorted with respect to their assigned priority. The priority can be low, normal or high respectively. EDTs with a high priority are ranked first of course. Similarly the EDTs with a low priority are added to the end of the queue. Future implementations are supposed to use available schedulers of the underlying OS, such as the Earliest Deadline First (EDF) scheduler of REFLEX [70], to provide a more precise and fair scheduling.

On dispatching an EDT into the *EDT evaluation* for execution, the sensing devices are triggered to deliver actual sensor readings required to decide about the existence of the described phenomenon by evaluation of the EDT. In case of a positive evaluation result, either a voting is triggered or the *Handler box* is called to execute respective associated handle methods. The *EDT evaluation* also manages collaboration with other sensor node if necessary. Details about collaboration are given later in this Chapter.

Finally, the timer of the EDT is set up to the next evaluation interval and the EDT is returned to the *EDT scheduler*. In addition to the configured evaluation interval, EDTs are also be triggered on demand by voting and collaboration requests from other devices. In that case, *EDT evaluation* is executed as usual but consequently triggers answering the request, too. After finishing an unscheduled evaluation, the corresponding timer is set to a full detection interval again. On the one hand, that assures a sufficient detection interval as required by the specification. On the other hand, it simultaneously reduces the number of evaluation processes to a certain minimum to save energy resources.

4.2 Establishing Event Decision Tree

Event descriptions are parsed at the sensor nodes to generate evaluable event configurations. Therefore the sensor node establishes an Event Decision Tree (EDT) representing the event based on the sensor data element of the event description. The EDT enables every node to self-divide event queries according to its resources and to execute the complete event evaluation process. Unlike in other approaches, nodes are not only used as data source for sensing and

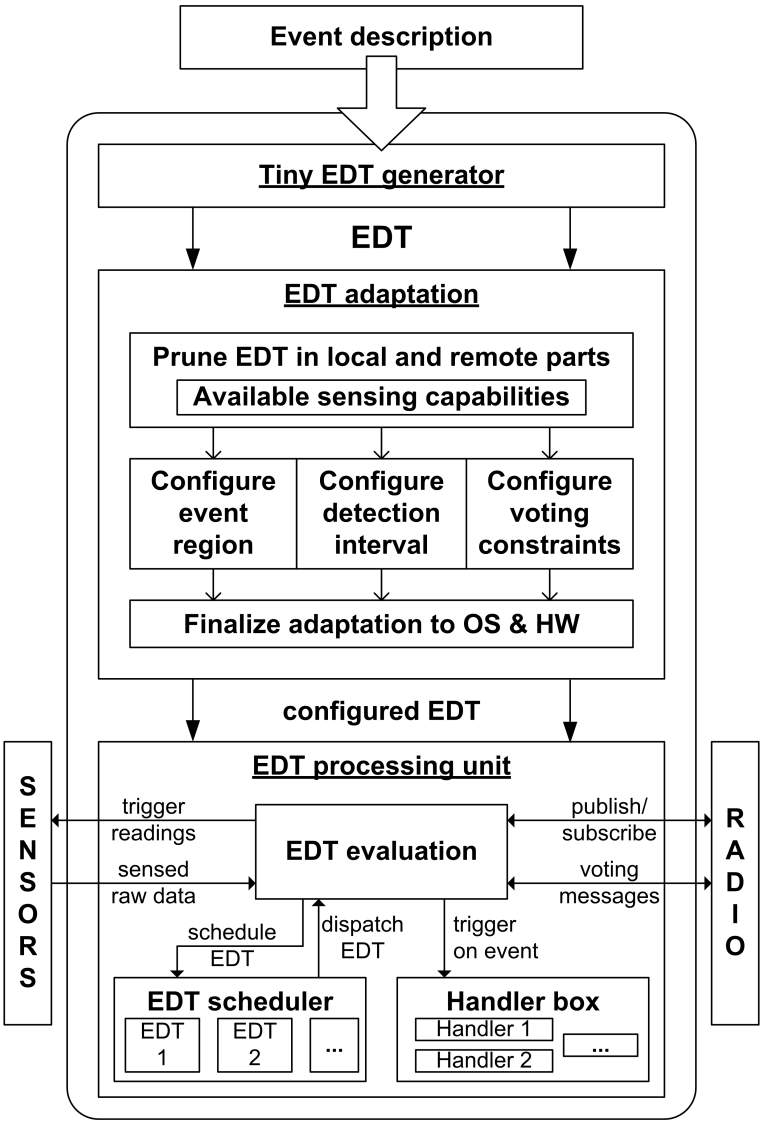


Figure 4.1: Detailed architecture of the EDT-engine.

distributing raw data. In fact, every node can independently analyse and process its sensor readings and come to a final Boolean decision about the occurrence of events. The EDT is a fully distributed concept that does not require special nodes for information-fusion and final evaluation. This is considered mandatory to prevent from SPoFs, which are naturally arising if only one or a few nodes are enabled to execute the complete detection and evaluation process. Such concept further significantly reduces the energy consumption in contrast to other approaches by omitting to distribute sensed data at each detection interval.

A tiny EDT-generator based on a Generating Finite State Machine (GFSM) was developed. Since the graphical representation of this state machine is confus-

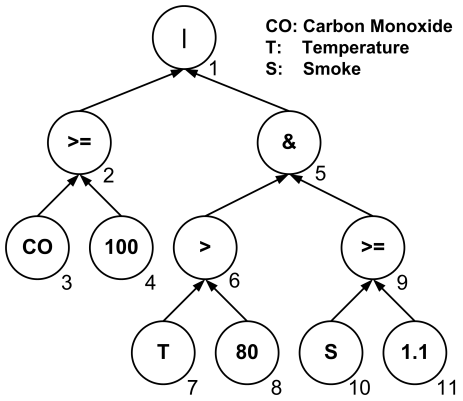


Figure 4.2: Pre-ordered EDT of the fire detection example.

ing due to the number of transitions, the respective state transition table is given in appendix A.3. The complete GFSM indeed has nine states but the last state is the stop state and provides no further functionality. Hence, the stop state was not implement in the EDT-engine, which gets by with eight states. The parsing finishes either if the input was completely processed or if a current input would originally apply a transition to the stop state. That enables to implement the EDT-generator on almost every available sensor platform. In the simulator the implementation of the GFSM required only 25 lines of C/C++ code. The EDT-generator transforms the prefix notation of the sensor data element into a congruent representation as an EDT. The EDT consists of leaf nodes, which identify sensing capabilities or constant values according to the specification of thresholds. The leaf nodes are child nodes of relational elements. Those nodes constitute the respective relation of its two children. These minimal trees of three nodes are primitive events. In complex events they become respective subtrees.

Logic nodes, representing the logic combinations of several primitive events, are generated as parent nodes on top of relational elements. In the fire detection example, the root node of the EDT represents a Boolean OR-relation of the thresholds regarding the carbon monoxide and combination of smoke and temperature. The equivalent EDT is depicted in Figure 4.2. For further processing the tree nodes are pre-order numbered during their creation from the event description. That assures the same initial tree labelling on every device in the network, which is necessary for efficient exchange of event information later.

4.2.1 Evaluation of the EDT

An event evaluation procedure is either triggered by internal event-related timing constraints, which are specified in the execution element or by evaluation requests from other devices. The EDT can be evaluated automatically in a bottom-up manner starting from the leaf nodes in order to determine a Boolean value at

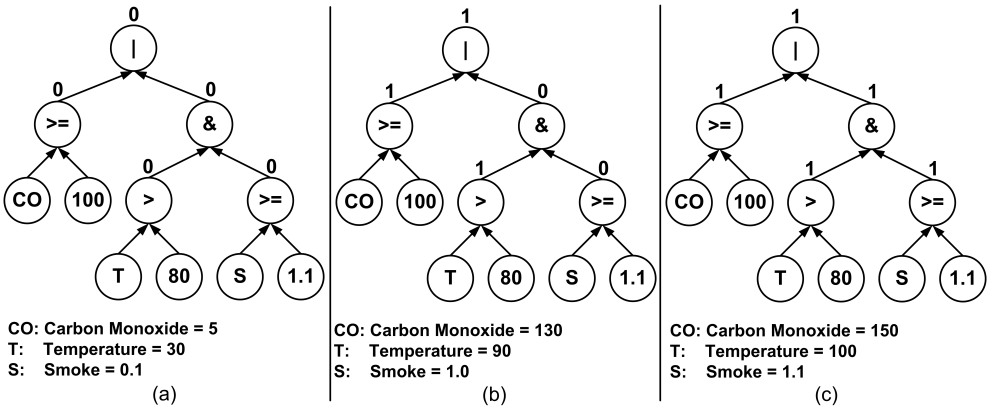


Figure 4.3: Possible evaluation results of the fire detection EDT. Depending on different actual sensor readings, Boolean values are assigned to the tree nodes, which are depicted as numbers on top of these. In (a), no threshold is exceeded and hence, the final evaluation result is 0/FALSE. Since the carbon monoxide threshold is exceeded in (b) and all thresholds are exceeded in (c), the corresponding root nodes evaluate to 1/TRUE, which is a positive detection result.

the root node, i.e., the final event evaluation result. All EDT nodes representing sensing capabilities are assigned with actual sensor readings. In the given example, these are the *CO*-labelled node for carbon monoxide, the *T*-labelled node for temperature and the node *S*-labelled for smoke readings. Afterwards the EDT is evaluated by comparing the children of all nodes according to the operation defined at the parent node. As a result, Boolean values are assigned to relational and logic nodes. If the value of the EDT-root node evaluates to TRUE, i.e., the event was detected, all specified event handlers are triggered for further processing. A new EDT evaluation is triggered when the next assigned evaluation time is reached. To illustrate this process, Figure 4.3 depicts different evaluation results of the example EDT. Depending on different assumed sensor readings, the Boolean values of the tree nodes are presented. Whereas the evaluation result in (a) is negative, the root nodes of (b) and (c) state a positive result due to the exceedance of the carbon monoxide threshold (b) and both thresholds of smoke and temperature (c) as well.

4.3 Local adaptation of EDTs by pruning

Up to here, it was assumed that sensor nodes possess all sensing capabilities to evaluate the complete EDT itself. If that assumption cannot be granted, local detection of events becomes impossible without collaboration with other sensor nodes possessing the required capabilities. The EDT-engine additionally enables sensor nodes to evaluate events and respective EDT even if they provide only a

subset or even no sensing capability. Such a lack of capabilities could be either by design or by failed sensing units. Hence, certain branches or subtrees of the EDT cannot be evaluated by the node itself. In that case, sensor nodes need to collaborate to exchange event information.

The exchange of sensed raw data, which is done by most approaches, is very inefficient from two points of view. First, permanent exchange of sensor readings leads to a huge number of transmissions and hence, consumes much energy and reduces network performance. Second, transmitting raw sensor data requires to use rather large data packages, depending on the number of readings and their accuracy, i.e., the size of each value usually varies from two to four bytes. Since a conceptual main goal is to remain very energy efficient, the focus is on minimizing the number of transmissions and the amount of data to be exchanged. Instead of exchanging raw sensor readings at each detection interval, sensor readings are locally processed first and finally one bit is submitted only, which is the Boolean value of a particular EDT-node. There already exist approaches that share information in a "yes" or "no" style, e.g., in [35], but these can only state the final complete detection result. This concept focusses on efficiently sharing information about both, complete and partial events.

In case of using EDT, the Boolean value of only one particular EDT-node has to be transferred. Missing node values may be delivered by neighbouring nodes that share the specified region of event. To prepare these data exchanges, every sensor node has to determine which EDT-node information is missing at the local EDT. Therefore the following algorithm prunes the established EDT until it contains the minimum required EDT for local event processing:

1. Mark each leave as pruned that represents an unsupported sensing capability.

2. Search all nodes that possess at least one marked child excluding the root node[1].

   2.1 Mark node as pruned, if

   a It represents a mathematical operation or

   b The unmarked child represents a constant or

   c All child nodes are marked as pruned.

3. Repeat step 2 until no new nodes are marked. After that, all undecidable subtrees are marked.

4. Prune all marked nodes except for the root nodes of the marked subtrees.

5. Declare all left marked nodes as "undecidable".

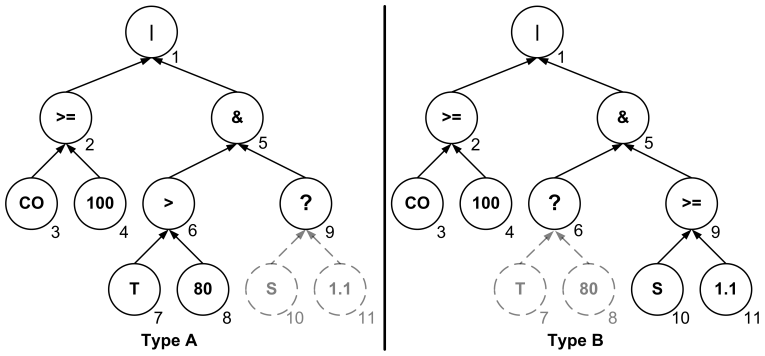


Figure 4.4: Pruned EDTs for two different types of sensor nodes monitoring the introduced *fire* event. Nodes of type A provide sensing facilities for carbon monoxide and temperature whereas nodes of type B provide sensing facilities for carbon monoxide and smoke. Consequently, each type of node prunes a certain part of the EDT that cannot be evaluated locally. Resulting "undecidable" nodes are labelled with "?". Hence, the Boolean values of these nodes must be obtained from other nodes in the specified region of event.

After pruning, EDTs may contain nodes, which are marked as "undecidable". Respective Boolean node values must be obtained by other nodes in the region of event. Let us assume to use two different types of nodes (A and B) for the introduced fire detection example. Nodes of type A provide carbon monoxide and temperature sensors whereas type B nodes provide sensing facilities for carbon monoxide and smoke. Hence, the initial EDTs generated at these nodes must be pruned with respect to available sensing capabilities.

Accordingly, type A nodes cut the branch containing the smoke readings and type B nodes respectively cut the branch containing the temperature readings. That results in two different EDTs at the sensing devices, each containing one node marked as "undecidable". Thus, type A nodes require information about tree node number 9 whereas type B nodes require information about tree node number 6. Both resulting EDTs are displayed in Figure 4.4. At regular evaluation, the EDT also checks the status of the sensing devices. In the case that sensing devices fail during application the sensor node runs the pruning again to locally self-adapt the EDT to the current situation. In addition, sensing devices may fail transiently only. In that case, the sensing device again is available and hence, the EDT can be reconstructed into its original form by removing the "undecidable" marking from respective EDT-nodes.

By pruning, the EDT may degenerate to a minimal tree consisting of the root node with "undecidable" children. Such an EDT enables sensor nodes that possess no suitable sensing capability for event detection, to serve as a "bridge".

[1]Since an EDT is a binary tree, every node possesses at most two child nodes. Hence, either one or both child nodes are marked as pruned in that case.

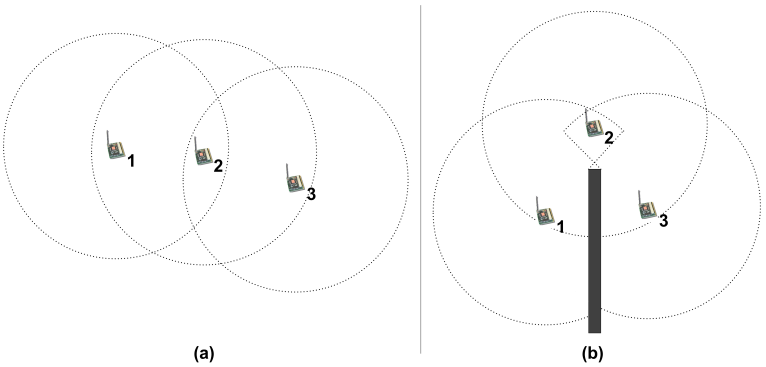


Figure 4.5: Example deployments that may require node *2* to serve as a bridge for the nodes *1* and *3*. In (a) the nodes *1* and *3* do not share an event region due to their distance . In (b) these nodes share their regions indeed, but cannot communicate directly due to an obstacle between them.

These nodes are of interest if they are located between two or more nodes that possess the required sensing capabilities but cannot communicate directly or do not share the same event region. The only prerequisite is that these nodes share the event region of the bridge node. Figure 4.5 displays example deployments for both cases. Here node *2* shares its event region with the nodes *1* and *3* and hence, may perform the bridging functionality for these nodes. In Figure 4.5(a) the nodes *1* and *3* do not share an event region due to their distance. In Figure 4.5(b) these nodes share their regions indeed, but cannot communicate directly due to an obstacle between them. In such a scenario all participating nodes deliver their parts of event information to the bridge node, which is finally enabled to decide about the occurrence of that event. After having identified the "undecidable" parts for event detection on each sensor node, those have to efficiently share necessary information. A suitable collaboration scheme maintaining this data exchange is presented in the next Section.

4.4 Collaborative exchange of event information

To save energy resources, wireless sensor nodes should communicate if and only if it is absolutely necessary. A suitable collaboration mechanism in sensor networks must further self-adapt to changing network situations and consider application requirements. In particular, the following questions are of primary concern:

Is there a need to transmit some event information?

If yes, which node has to transmit what information?

Is there really a sensor node that receives the data?

Additionally, the amount of exchanged data ought to be kept as small as possible. This Section presents an adaptive and easy-to-scale mechanism to efficiently share event information based on a publish/subscribe approach.

4.4.1 Publish/subscribe scheme

It is quite obvious that request-ACKnowledgement-based (ACK) communication schemes can be used for reliable collaboration indeed, but produce a huge amount of traffic and are therefore inefficient for sensor networks. In idealised scenarios, the ACK-based data exchange requires at least two transmissions per detection interval and node, i.e., one request and one acknowledgement message. In usual application such a request is spread as broadcast and hence, several nodes may answer to particular request. To simplify matters, this possibility is ignored but would of course further increase the required traffic for ACK-based data exchange. The publish/subscribe approach maintains the exchange of EDT-node values and reduces the traffic by submitting only changes of node values to achieve longer time intervals without any transmission. The focus is on reducing the package payload for application data as well as the number of transmissions. The estimation and the simulations to be presented show that the proposed publish/subscribe scheme also outperforms such idealised ACK-based variants.

Minimizing the amount of exchanged data Since every bit to be transmitted is expensive with regard to energy consumption, the amount of exchanged data has to be minimised. The EDT shares event information efficiently using a few bytes only. In contrast to existing approaches that need to share raw sensor data, which is usually between two and four byte per value plus identifier, here only the Boolean value of a certain EDT-node is of interest. Thus, a data transmission has to contain only the event identifier, the number of the respective EDT-node and the current Boolean value assigned to that node. Please remember, prefix-numbering the complete EDT before pruning assures that the EDT-nodes at each device in the network possess the same numbering. Such labelling scheme allows to efficiently describe the node of interest and the assigned value with one byte only. That byte consists of one bit representing the Boolean value and seven bits representing the address (number) of the EDT-node. Hence, 128 different nodes in one EDT can be addressed. If an EDT contains more than 128 nodes, an extra byte for addressing is used.

In addition, the event identifier must be submitted given that a sensor node is enabled to configure several events and respective EDTs concurrently. To simplify matters, here a readable event identifier *fire* is used for the example messages given in Figure 4.6. These messages illustrate the data payload for collaboration based on the respective pruned EDT of node types A and B. If the event identifier is chosen to be a unique number less than 256, e.g., this

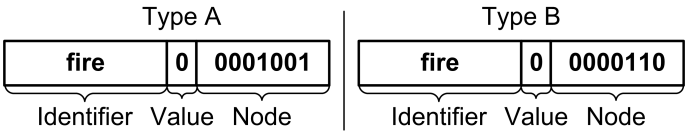


Figure 4.6: Example of collaboration messages for type A and type B nodes in the fire detection scenario. According to their customised EDTs, nodes of type A request information about EDT-node number nine whereas nodes of type B request information about EDT-node number six. Please note, here the labels of EDT-nodes are given in binary notation.

number may be generated while preparing the event description, all necessary information can be transmitted by two bytes only. Hence, this scheme reduces the required data payload for collaborative exchange by at least 50 percent. Please note, this payload structure is used for requests and responses as well, which are differentiated through the identifier of the transmitted package.

Subscribe a data interest Missing values of "undecidable" EDT-nodes must be obtained by suitable other sources as mentioned, e.g., neighbouring nodes. Thus, the sensor node broadcasts a data interest (subscription) into the network to find suitable information suppliers. If the event specifies a region of event, this subscription must also contain the location of the subscribing sensor node. On receiving a subscription, the sensor node compares the location data to determine whether both nodes share the region of event. Only if that holds true or if the request contains no location data, i.e., the one hop neighbourhood event region, the received subscription is of interest. The receiving sensor node searches its own respective EDT to determine whether it can provide the requested information. The requested EDT-node is marked with a "toPublish" flag and the sensor node answers the request by providing the current value of the requested EDT-node. In all other cases, the node discards the received subscription without further processing. Subscriptions can also consist of many concurrent data interests in case of requiring information about several "undecidable" EDT-nodes of one or more events. That significantly reduces processing and communication effort required for packaging, addressing, transmission etc.

Publishing EDT-node information On event evaluation the current state of each EDT-node is determined. Results at nodes marked with the "toPublish" flag are also important for other devices in the network and hence, ought to be published. To save resources these evaluation results are not transmitted periodically. Only first-time subscriptions and state changes require transmission of the current node state. If a device accepts a received subscription for the first time, it answers with the current node state to provide an initial value for the subscriber. Since a node

state is of Boolean type, only state changes must be submitted to update the node state at the subscriber. If node states change rarely, the number of required publications is significantly reduced. Even in the worst case, i.e., the node state changes at each evaluation, this scheme requires the same overhead as usual methods where values are transmitted repeatedly at every evaluation period.

Adaptiveness of publish/subscribe Using a publish/subscribe scheme is rather simple if reliable communication architectures and fixed network structures are provided, but WSNs are subject to unpredictable behavior triggered by sudden changes in context, connectivity, working mode etc. That especially holds true if mobility of nodes is provided. To ensure a certain level of reliability and efficiency, some basic essentials have to be considered from respective points of view of subscribers and publishers. How does the subscribing node know, whether some other node received the subscription, accepted it or is still providing publications? On the other side, the publishing node requires to know whether there is still a subscriber requiring event information. These problems could indeed be solved by using a simple ACK-scheme or timers for every transmission to inform the sender about the success. Unfortunately, both methods are inefficient for WSNs as mentioned. Furthermore, publish/subscribe is designed to achieve large periods without any transmission, which is not possible with ACK-schemes or timers.

Due to assumed conditions, the publish/subscribe scheme must adapt frequently to reach a certain level of reliability in event detection. Certainly, the overhead needed for adaptation must be kept as small as possible but still allow for balancing the adaptiveness with respect to the application provided. Accordingly, publications and subscriptions should either be removable or be valid for certain time periods only. The latter is much more suitable for WSNs where unforeseen changes leave no chance for appropriate responses or un-subscriptions. Therefore an adaptive lease procedure limits the validity of publications and subscriptions. It allows to subscribe a data interest for a certain lease period only, after which the publish/subscribe relation has to be renewed. Such lease-based publish/subscribe requires significantly less transmissions than ACK-based variants and enables event-assigned lease intervals.

4.4.2 Lease procedure

A lease-based subscription specifies a certain time interval determining the validity period of subscriptions during which associated publications have to be sent. This lease period is determined as the product of the event-assigned lease factor and the event evaluation interval. Both factors are given by the event specification. The lease factor allows the user to adapt the lease period to the monitored event and to the expected conditions in sensor networks. It enables fine-tuned and customised lease intervals. For example, sensor networks which are subject

to permanently changing situations or node mobility require a high adaptiveness. Those should apply short lease intervals. In contrast to that, sensor networks deployed at rather fixed network structures could make use of larger lease intervals to save energy and extend the overall network lifetime. Please note, if the lease factor is chosen to be one, i.e., the leasing time is one detection interval, this scheme converges to ACK-based approaches.

On receiving a subscription, the node determines the expiration date of the respective publication. The expiration date is assigned to the corresponding EDT-node together with the "toPublish" flag. After initially publishing the current EDT-node value, any further change is published as long as the "toPublish" flag is set. Consequently, the flag is automatically removed from the EDT-node when the expiration date is reached, i.e., the lease has expired. Similar to the "toPublish" flag, the subscriber assigns an expiration date to the requested "undecidable" EDT-node. Even if no publisher responds to the subscription, the node sends no new subscription before this expiration date has expired. That assures to renew the publish/subscribe relations with respect to the configured adaptation rate only. Other approaches usually try to subscribe at each detection interval again, which heavily drains the power resources. Usage of similar lease-based approaches is also well known in other application areas, e.g., for labelling of references and objects in automatic garbage collection [28] or for allocation of resources like the Internet Protocol (IP) addresses from servers using the Dynamic Host Configuration Protocol (DHCP).

To save more energy, new and renewed leases are distinguished to save the initial respond of the publisher too, since it is not necessary if no change has occurred. If earlier agreed leases are to be renewed only, the publisher does not respond with the initial node value but extends the lease period and continues providing state changes until the newly assigned expiration date is reached. In addition, publisher and subscriber renew the lease period automatically upon notification of a state change. When publishing data during a valid lease, the expiration date of the "toPublish" flag is set to a full lease period again. Similarly, the subscribing node renews the expiration date of a current subscription when receiving a respective publication.

Figure 4.7 displays sequence charts of both lease extension cases as well as the ACK-based scheme for comparison. In the ACK-based variant, which is displayed in (a), the subscriber requests event information at each detection interval, which is accordingly responded by the publishing node. In contrast to that, (b) and (c) illustrate the lease allocation in case of no event (b) and event (c) while applying a lease factor of three. Both cases require to provide the current node value by an initial publication of course. In case of no event (b), the subscription is renewed by the subscriber after the lease has expired. If an event occurs during a valid lease (c), the lease period is automatically extended on both sides via the

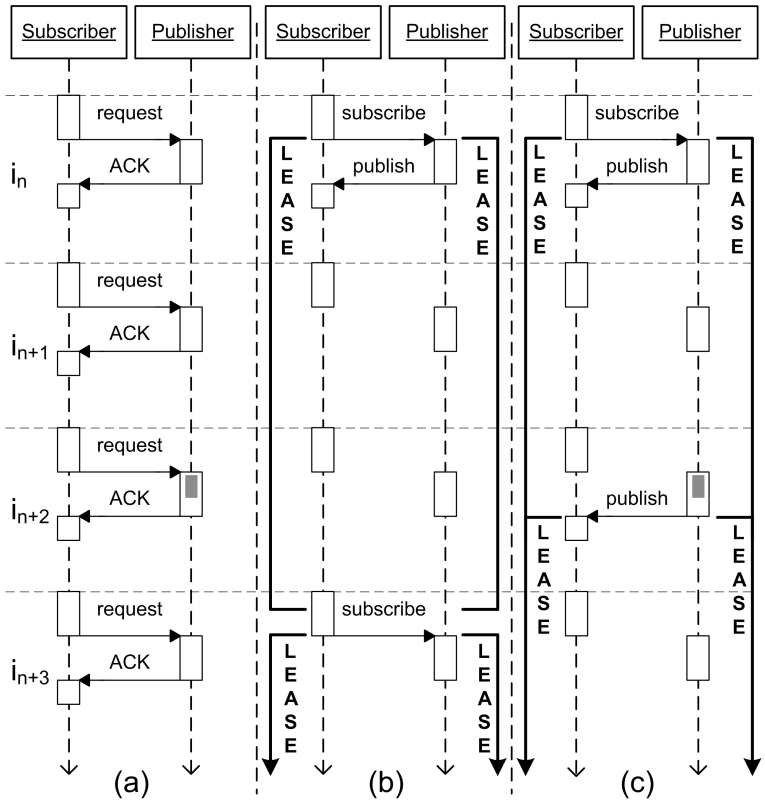


Figure 4.7: Sequence of information exchange between a single subscriber and a single publisher during four detection intervals i_n to i_{n+3}. (a) displays the performance of the ACK-based variant, which is always equal regardless of the existence of events. (b) and (c) illustrate the lease allocation applying a lease factor of three, i.e., the subscription is valid for three evaluation intervals. In case of no event, see (b), the subscription is renewed by the subscriber whereas the existence of events allows to extend the lease on both sides via the publication message.

publication message. In such simple scenario, the lease-based approach already saves more than 60 percent of messages.

Since a lease can be automatically extended by publications without a respective acknowledgement message from the subscriber, there is a risk that the publisher side runs into a kind of infinite loop. In other words, after publishing the initial node value, the publisher may renew the lease and the "toPublish" flag again and again, while the subscriber disappeared in the meantime. To cope with that, an exceptional termination condition for publications was integrated. The publisher counts the number of automatic lease renewals and removes the "toPublish" flag, when the value of the counter equals the given lease factor. Hence, that exceptional condition forces the subscriber to renew the subscription again if it still requires information about the respective node value. Conse-

quently, each subscription resets the counter at the publisher to zero.

It is quite obvious that this communication scheme is well suited for low power applications such as environmental and structural health monitoring. Besides such high efficiency in power consumption, another main goal of this work is high reliable event detection, which consequently requires a stable and reliable communication scheme, too. Therefore the introduced approach can operate in a reliable mode as well. The reliable mode combines the advantages of the introduced publish/subscribe scheme with the reliability benefits provided by ACK-based communication. The reliable mode introduces retransmissions on the application level. Usually retransmissions in case of message loss can be assumed to be part of the Medium Access Control (MAC) layer. If retransmissions are not provided by the MAC protocol or the link reliability of the underlying network is unknown, the publish/subscribe scheme can provide a similar feature on the application level.

The user can activate the reliable mode by setting the "reliableMode" attribute in the <EVENT> element of the event specification. The reliable mode applies the introduced lease-based publish/subscribe scheme, too, but enforces to explicitly acknowledge every transmitted data packet. If the ACK-message fails while the reliable mode is enabled, the already renewed lease is removed immediately. A publication in response to an initial subscription is implicitly used as ACK-message, too. This introduces a little overhead indeed, but still outperforms usual ACK-based variants. Figure 4.8 illustrates the performance of the reliable mode when applied to the same scenario shown before in Figure 4.7. Similarly, it compares the behaviour of the ACK-based variant (a) to the cases of no event (b) and event (c) in the publish/subscribe approach. Even here, the reliable mode still saves 50 percent of required messages.

Finally please note, the reliable mode can be applied to a specific event and hence, allows to customise the used communication scheme for each event configuration. In contrast to other approaches where the communication protocol is identical for all configured tasks, here the EDT-engine may execute both modes simultaneously depending on the configured events. To summarise, it is a simple fact that in theory the lease-based publish/subscribe provides a considerable benefit with respect to the number of required transmissions, even if the applied lease interval is rather short. Nevertheless, this has to be further proved by traffic estimation and simulations as well, which are presented in the next Sections.

Efficiency estimation

In order to reinforce the efficiency and theoretical advantages of using the lease-based publish/subscribe distribution of event information, the costs of the introduced lease procedure are compared to an idealised ACK-based communication scheme. The required traffic is analysed considering the respective points of view

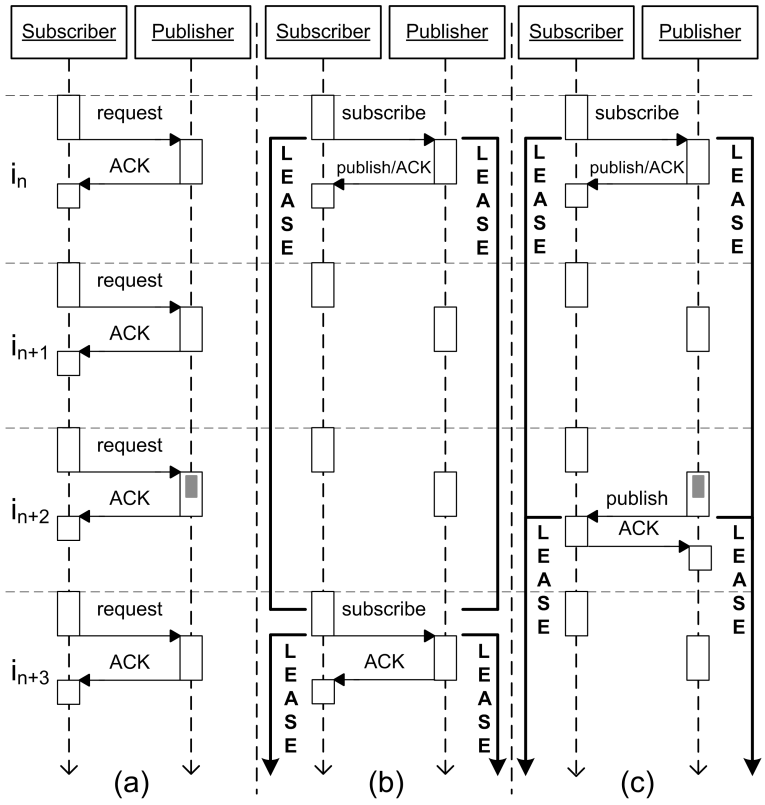


Figure 4.8: Sequence of information exchange for the same scenario as shown before in Figure 4.7 while applying the reliable mode for the lease-based publish/-subscribe scheme in (b) and (c). (a) displays the performance of the ACK-based variant for comparison. Even with an enabled reliable mode, the lease-based approach requires 50 percent less messages.

of a single subscriber and a single publisher and finally estimated for the entire WSN. Additionally, the estimation shows that there exists a break-even point for the benefit of the lease procedure in comparison to the ACK-based scheme for almost all scenarios. The worst-case scenario, which exists in theory only, results in marginal overhead for the lease procedure. Details on the estimations and a diagram can be found below in this Section.

The following examines the efforts from the application point of view without considering MAC and physical layer issues. It is quite clear that wireless communication depends on many more parameters than on the applications running. Many published projects have proven that links in WSNs are unreliable [24, 56, 60, 75]. According to this, the number of originally required messages in the network increases by a certain amount. This is a clear fact, but since two different communication schemes under same network conditions are compared, both would assign such traffic increase. Thus, to simplify the estimation,

i	Event evaluation interval
T	Number of consecutive detection intervals
k	Lease factor ($k \geq 1$)
N_s	Number of subscribers in the entire network
N_p	Number of publishers in the entire network
p_t	Probability of changes of EDT-node values ($p_t \leq 1$)
n_s	Average number of subscribers for a single publisher

Table 4.1: Parameters and terminology used for efficiency estimation.

unreliable links are not considered for direct complexity comparison and hence, the essential required traffic is calculated only. Consequently, the approach that performs better in the idealised network condition case, is expected to do so with unreliable links, too. Of course, less traffic will be also less affected by unreliable links. Much traffic additionally increase the possibility of link failures and message collisions in the network and hence, increases the total message loss.

The simplest way to assure reliable collaboration among sensor nodes is to use ACK-based communication. To provide a proof of efficiency, here a best case scenario assumption of the ACK-based scheme is made. That is, it requires exactly two messages per event detection interval and node. The best case neither regards that several publishers may answer to the same request nor that several subscribers may acknowledge the same responded value simultaneously. Obviously, both cases would increase the required traffic. Explicitly confirming each request or responded value establishes a form of bilateral relationship. It directly informs the subscriber that there really exists a suitable publisher and that one gets a feedback that there is still some subscriber requiring information. Disappearing subscribers or publishers can be recognised immediately, which is the most important advantage of the ACK-based variants. The drawback is, that every data exchange requires at least two transmissions per evaluation interval to reconfirm the relationship. According to this, a subscriber either sends one subscription or one acknowledgment for received publications per detection interval. Publishing nodes respectively publish data as subscribed or answer new subscription requests by one message per detection interval i. Expression (4.1) determines the required messages within a consecutive sequence of detection intervals for distributing event information using an ACK-based communication scheme. The number of consecutive detection intervals is represented by T. Table 4.1 lists the parameters used for traffic estimation.

$$T * (N_s + N_p);\ T \in \mathbb{N} \tag{4.1}$$

In contrast to that, a lease-based approach eases the strong relation of ACK-based communication in favour of less overhead. Due to the fact that the validity of a subscription extends automatically, the lease procedure assures that pub-

lishing event information is performed if and only if it is necessary. That adapts publications to subscriber requirements periodically and saves a lot of energy at the publisher side. Therefore each subscription and publication assigns an expiration date e, determined by the event-assigned lease factor k and the number of event evaluation intervals i to be leased, see Equation (4.2).

$$e = i * k;\ k \geq 1 \tag{4.2}$$

With regard to the given lease factor k, a subscriber does not send a subscription message at every detection interval but after every k intervals. To inform all suitable publisher at once, a subscription is broadcasted. Within a consecutive sequence of detection intervals T this results in $\frac{T}{k}$ messages. A publisher sends a one-time message to all new subscribers (n_s) to provide the initial node value. Afterwards a publisher notifies its subscriber(s) if and only if the Boolean node value changes. The probability of such change is considered p_t, resulting in $(T-1)p_t$ total publications. Due to the exceptional termination condition for publications, one may argue that there exists a chance of unnecessary publishing data, if the last associated subscriber disappears during an active lease. Hence, the publisher may continue with publishing data and automatically renews the lease for at most k times. This chance exists indeed but since $(T-1)p_t$ considers publishing data all the time, that case is already included. The total required traffic for a lease-based publish/subscribe is calculated using Expression (4.3). Here each publisher is assumed to definitely publish data as expected by p_t. It is not included that a potential publisher may have no associated subscriber, which would result in less required messages of course.

$$N_s\frac{T}{k} + N_p((T-1)p_t + n_s);\ p_t \leq 1; T \geq k \tag{4.3}$$

To compare both approaches, these are analysed from the points of view of subscribers and publishers. With regard to subscribers the lease-based approach clearly outperforms the ACK-based variant, see Equation (4.4). Even considering the worst-case, i.e., a lease extends after every interval ($k = 1$), results in equal cost of communication.

$$TN_s \geq \frac{T}{k}N_s \ \Leftrightarrow\ T \geq \frac{T}{k} \ \Leftrightarrow\ 1 \leq k \tag{4.4}$$

From the view of publishers the cost analysis and comparison requires evidencing the validity of Equation (4.5), which is equivalent to (4.6). To summarise, there exists a T that satisfies Equation (4.6), unless $p_t = 1$. In other words, after a certain number of intervals even here the lease-based procedure performs better than the ACK-based. Only in case of publishing data at every interval because of permanently toggling events $p_t = 1$, the lease procedure requires a $(n_s - 1)$ more messages than the ACK-based scheme, which provide the initial value. With

growing number of intervals, that overhead becomes nearly irrelevant. Moreover, the probability of $p_t = 1$ is not existent in real applications and therefore becomes negligible. Event detection is applied to monitor events that occur rather rarely if at all, which in turn mean an event detection rate of 1 or even close to 1 is not existing in such a system. If such behaviour is detected, it merely indicates a "wrong design" or "broken sensors" than the event to be detected. Event-based detection is usually designed to have an event probability of less than 50% in average. For example, defining an event to be triggered as *temperature* $\geq$ 5 *centigrade* in a house will usually results in frequently triggered events by the sensor node. In such a scenario the event probability will be higher than 90%. To gather an equivalent information, the event definition is to be reversed. Hence, the sensor network should be reconfigured to trigger an event in case of *temperature* $<$ 5 *centigrade*. This provides a similar information but would result in an event probability of less than 10% in the given example.

$$TN_p \geq N_p((T-1)p_t + n_s) \tag{4.5}$$

$$\frac{T - n_s}{T - 1} \geq p_t \tag{4.6}$$

Finally, a break even point analysis for the entire network is carried out to show that the lease-based procedure always outperforms the ACK-based approach after a certain number of intervals. Therefore the validity of Equation (4.7) has to be proven. The break-even point for the lease-based scheme can be easily determined by transformation to T, see Equations (4.8) and (4.9). As it is easy to see, Equation (4.9) is solvable except if $k = 1$ and $p_t = 1$ at the same time. Whereas choosing $k = 1$ is possible but not reasonable, the case of $p_t = 1$ is rather unlikely as mentioned. In summary, there always exists a break even point for the benefit of the lease-based publish/subscribe approach, from which it outperforms even idealised ACK-based methods.

$$T(N_s + N_p) \geq N_s\frac{T}{k} + N_p((T-1)p_t + n_s) \tag{4.7}$$

$$TN_s - N_s\frac{T}{k} + TN_p - TN_pp_t \geq N_p(-p_t + n_s) \tag{4.8}$$

$$T \geq \frac{N_p(n_s - p_t)}{N_s(\frac{k-1}{k}) + N_p(1 - p_t)} \tag{4.9}$$

To illustrate the theoretical analysis, the introduced fire detection scenario was applied to estimate the costs of both approaches using a WSN with different event-defined lease factors k. The assumed sensor network consisted of 100 nodes, providing 50 type A nodes measuring carbon monoxide and temperature as well as 50 type B nodes measuring carbon monoxide and smoke. Since the detection of the event *fire* requires carbon monoxide, smoke and temperature readings, all nodes took the roles of subscriber and publisher at the same time, i.e.,

$N_s = N_p = 100$. The nodes were uniformly distributed in a field of 25x25 meters. According to the introduced event specification, collaborative detection of the *fire* event ranges over a distance of five meters based on the given radius of 2.5 meters around the related sensor node. Thus, each publisher was assumed to be associated with an average of three subscriber ($n_s = 3$). Finally the number of probable events was defined, which determine sensor readings that exceed or fall below the specified threshold. For demonstration, an event probability of ten percent $p_t = 0,1$ was applied. Please note, the event probability is certainly lower in real fire detection scenarios but here it fits well to verify the benefits of the introduced approach. A realistic probability for a fire in a aeroplane would be less than 0,001 percent for example. That would of course again result in much less traffic for the lease-based scheme.

The diagram in Figure (4.9) represents the comparison of estimated traffic for the entire network using an ACK-based variant and the lease-based approach assigning different lease factors k. Remember, since each detection interval in T represented ten seconds in lifetime, $k = 6$ means renew of leases and adaptation of event detection was done every minute. The results clearly reinforce the efficiency of the lease-based procedure, which outperforms the ACK-based scheme yet after one minute ($T = 6$). It is quite obvious that the lease-procedure performs better with increasing the lease factor due to the reduced number of subscriptions needed. In the best estimated case, a lease period of ten minutes ($k = 60$) reduces the traffic in the entire network by a factor of 16 yet after three hours ($T = 1080$). Finally, this is an estimation representing theoretical aspects only. Many projects reported that real deployments of WSNs might behave totally different as being expected by design.

4.5 Side effects of voting and collaboration

In spite of presented benefits of voting and collaborative event detection, there exist some side effects that must be considered separately. Collaboration obviously includes distributing partial event evaluation results to subscribing nodes to enable these to evaluate their respective EDT. Certainly, that may essentially effect the final evaluation results at these nodes. More precisely, the final evaluation result at subscribing nodes may highly or directly depend on received (published) values. Hence, several nodes may always generate the same final evaluation result as their local publisher. This is not of concern for usual detection, since otherwise subscribing nodes may possibly not have the ability to generate a final evaluation result at all. In that case, local event detection would depend on the results of the publishing node in either way.

In view of distributed event evaluation by voting, this case is quite of concern. Voting may either become useless or produce falsified voting results. If there exist

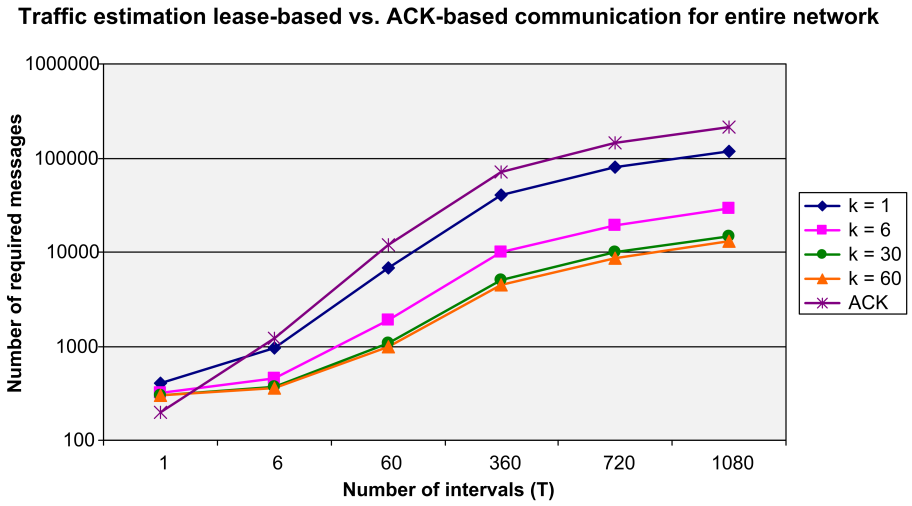


Figure 4.9: Comparison of estimated traffic for the entire network using an ACK-based variant and the lease-based approach assigning different lease factors k. The lease-based approach outperforms the ACK-based scheme yet after one minute ($T = 6$). In general, the lease-procedure performs better with increasing the lease factor due to the reduced number of subscriptions needed. In the best estimated case, it reduces the traffic in the entire network by a factor of 16. Please note, a logarithmic scale is applied.

no independent devices in that region, the voting result is always equal to the local results on the nodes. Hence, voting unnecessarily produces an overhead in traffic and energy consumption. In contrast to that, the publisher and assigned subscriber may wrongly overrule other independent devices if published values are faulty. That case is not detected by RMV and other known voting approaches and hence, requires to improve the voting algorithm and the evaluation scheme as well. The proposed solution is an extended evaluation scheme of the EDT to provide detailed local evaluation results, which finally affect the voting similar to the "confidence values" in TIBFIT [34]. Event evaluation at subscribing nodes further may not fully depend on published values. As presented in the example scenario using type A and type B nodes, these may either detect a fire depending on exchanged smoke and temperature values, or based on their carbon monoxide readings. Accordingly, all devices should rate results based on own readings higher than results based on published values to locally determines a weighted final evaluation result. A weighted RMV would allow to generate advanced voting results as well as to detect the cases in which voting is useless. However, a detailed evaluation increases the costs of processing, memory and communication. In addition, the associated overhead must again be customisable for each application. Hence, the costs and benefits of such approach must be well analysed before implementation. This is considered to be future work.

Of utmost importance in distributed systems like WSNs is the avoidance of fault propagation. In consequence of using the publish/subscribe collaboration scheme published faulty values could possibly propagate through the network from node to node. This happens when a value of an EDT-node, which was formerly obtained through subscriptions, is published further. In the worst case, faulty values could be distributed to all nodes in the entire network. Hence, stable or even reliable application becomes impossible. A simple condition gets rid of this problem. The implementation of EDTs must not allow EDT-nodes to assign values by subscriptions and to publish those values at the same time. In the presented implementation this is guaranteed by mutual exclusion of the "undecidable" and the "toPublish" flag. This assures, that faulty values cannot propagate beyond the defined event region of the publishing node.

Another side effect occurs if a subscriber has several publishers available at the same time. These may deliver different Boolean values for the same EDT-node within a certain detection interval. Hence, the subscriber has to determine the "correct" value of that EDT-node, or more precisely, the value that is most likely to be correct. To cope with that, the sensor node counts all TRUE and FALSE values per interval and finally comes to a majority decision. Certainly, the respective counters at each EDT-node are set to zero again after every evaluation run. In case of a tie situation, the last received value is taken.

A side condition that has not been considered so far but which is an important issue for distributed application, is the data security. This work does not focus on ensuring data security indeed, but it is an essential point in view of reliability, especially when considering mission- and safety-critical applications. Hence, some means providing data security must at least be mentioned. Using wireless communication increases the risk of malicious attacks because the sensor network becomes accessible from outside [74]. Whereas dozens of various attacks exist [32, 50], the integrity of voting and collaboration messages is of particular interest, to protect these against forging. Encryption is the key to provide confidentiality to the system. Based on that, the integrity of exchanged information can be assured by secure hashes [52]. These means can additionally be used on top of the presented event detection scheme, if necessary. Many different encryption methods are already available, but these quite differ in their strength and energy consumption depending on the algorithm and hardware used [51]. Just as mentioned before, the associated overhead is to be balanced between varying application requirements, too.

4.6 Performance evaluation

To give a proof of concept as well as to test the approach under various conditions a prototype of the EDT was implemented in the discrete event simulator

OMNeT++[2] [66] with an extension for simulation of WSN called Castalia[3] [53]. The introduced fire detection scenario was applied to WSNs that are subject to various failures in sensor readings and sensing devices. Based on that, the application of MV is compared to RMV and lease-based publish/subscribe approach is compared to the ACK-based variant. The simulation results are evaluated with regard to the detection accuracy and cost-efficiency of the introduced algorithms.

4.6.1 Evaluation methodology

Using a Boolean event detection allows to generate snapshots of the system state as a matrix. This matrix describes the sensor network by using the position data of the sensor nodes. It describes the respective x coordinates in the columns and the y coordinates in the rows. The matrix contains the actual or respectively last event detection results of all nodes, which are either *event* or *no event* respectively 1 or 0. As a start, the first run simulated the correct behaviour of the entire sensor network without any failures while snapshots from the system are frequently generated at every event detection interval. These snapshots represent the best case scenario and are used as the regular reference. In the following, all results and evaluations based on these snapshots are henceforth called ***reference***. Consequently, such snapshots were taken from all other simulation runs at equal simulation time, too. Finally, all snapshots of equal simulation time are matched against the regular reference to determine whether the sensor nodes in the simulation runs gathered the correct detection result or not. The simulation results are not only compared to the reference scenario. Each failure scenario was also executed without any fault tolerant improvement of voting or collaboration to gather the usual detection results of the sensor nodes. In the following, these runs are referred to as ***standard***.

The simulation results are analysed to determine the total detection accuracy, the number of required messages and the number of detected events and undetected ones and *False positives*. The total detection accuracy states the total number of correctly gathered positive and negative evaluation results per interval. Hence, a detection accuracy of 100% is given if all nodes within the phenomenon notify an event (positive result) while all other nodes do not register an event (negative result). As mentioned, the voting and the collaboration algorithms introduce a communication overhead, which is represented by the total number of sent overhead messages. Additionally, the average number of required messages per node and interval is determined to directly compare the cost-efficiency of all approaches. An overview of the simulation results is given in summary tables provided for each failure scenario. All simulation results are depicted as diagrams, too. For reason of clarity, these diagrams are separately provided in

[2] www.omnetpp.org
[3] http://castalia.npc.nicta.com.au/

Appendix B. The summary tables provide respective references. Here, only the most significant diagrams are directly referenced in the text.

Finally the number of detected events and detected phenomena are of interest. An ***event*** is triggered by a sensor node if the local detection result is positive. A ***phenomenon*** is the physical condition that is to be announced by events based on the generated sensor readings. Besides the total amount of detected events the number of *False positives* and the existence of undetected phenomena is of great importance. Especially in a mission and safety-critical context, such as the fire detection scenario, these numbers are essential to validate the reliability of the application. *False positives* are detected events in spite of inexistent phenomena and hence, these represent a false alarm that unnecessarily triggers further actions. Therefore and since *False positives* reduce the credibility of the system their numbers should be kept minimal. In view of the mission, undetected phenomena are much worse than *False positives*. Undetected phenomena are the most critical point because these represent a failed mission, e.g., an undetected fire. In the simulation scenarios a phenomenon is undetected (or missed) if no node detects an existing phenomenon. In other words, to satisfy the mission of fire detection it is sufficient to have at least one node stating the *fire* event.

4.6.2 Simulation parameters and deployment patterns

According to the introduced specification of the *fire* event, an event evaluation interval is ten seconds. The simulations ran three hours of simulated realtime, which is equivalent to 1080 event evaluation intervals in the fire detection scenario. For the rest of this Section, the simulation time is given in discrete time steps, i.e., the event evaluation intervals. Finally, the simulation parameters regarding the wireless communication need to be identified. Wireless communication may be subject to many restrictions resulting in an unreliable and sometimes nondeterministic performance. Many research projects already studied the parameters of link reliability, end to end delays, low power communication etc. These issues are indeed quite important, but are less considered in this work. To generate deterministic results for comparison, all simulation runs applied ideal conditions at the MAC layer and the wireless channel. Meanwhile, there also exists an approach that allows to deterministically simulate unreliable links in OMNet++ [69] but this work could not be considered at all.

The simulation results are taken from two different deployment patterns using a field of 22.5×22.5 meters containing 100 wireless sensor nodes. Each sensor node initially possesses all required sensing facilities, i.e., carbon monoxide, temperature and smoke detectors. Hence, all sensors are enabled to locally evaluate the complete EDT to gain local detection results. The first deployment using a uniform grid distribution is visualised in Figure 4.10. Due to the deterministically fixed positions of the nodes, this deployment enabled to check and analyse the

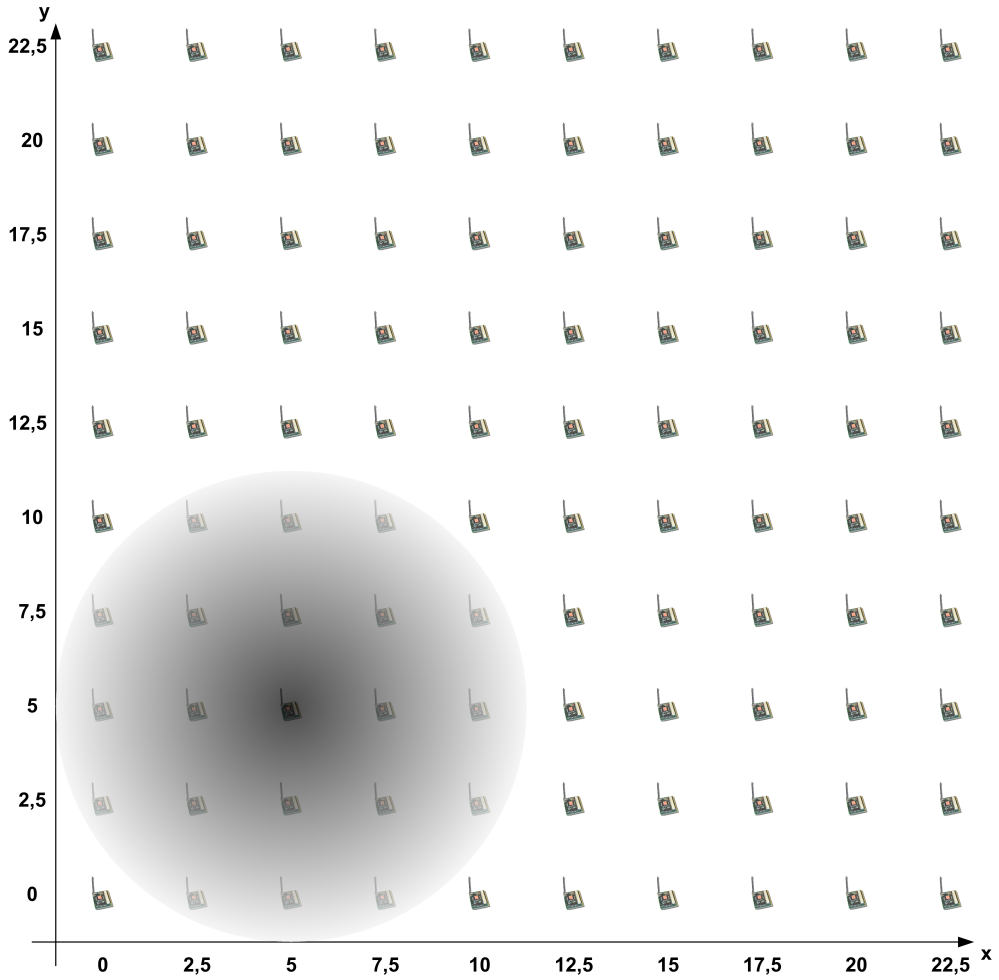


Figure 4.10: Initial deployment of 100 sensor nodes as uniform grid in a field of 22.5x22.5 meters as used for the simulations. The shadowed area symbolises the simulated deterministic phenomenon, which is specified in Figure 4.11.

correctness of the algorithms being applied to the grid scenario. For further comparison, ten different random uniform node deployments were generated. Each failure scenario was also applied to these deployments. Finally, the average results of all the simulation runs on each of the ten deployments is determined. These average results of the random uniform deployments are compared to the results of the grid scenario. Random uniform deployments may better represent the application of WSN in real world scenarios.

For deterministic event generation a simulated phenomenon is specified, which determines the actual sensor reading in a certain region. The simulated *fire* phenomenon had a circular dimension specifying high sensor readings in its centre, which decrease with the distance to the centre of event, see Figure 4.11. In particular, this phenomenon separates the network into four areas. The region

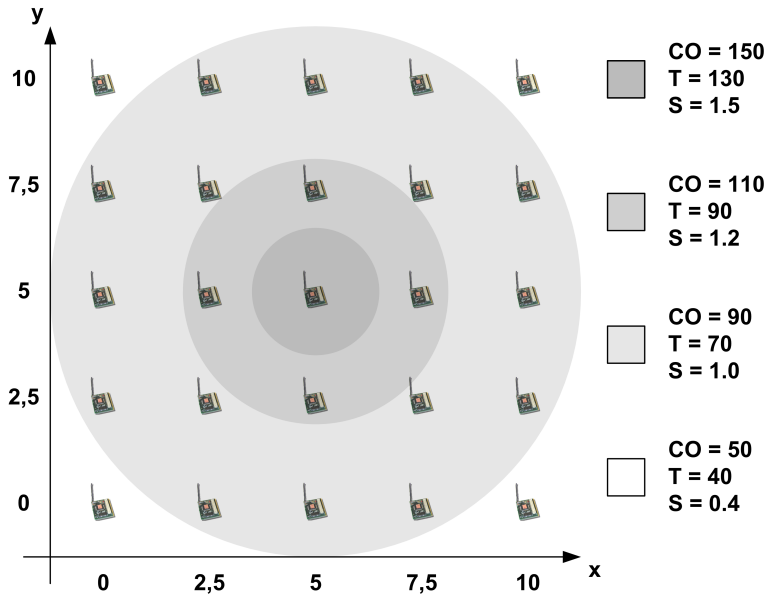


Figure 4.11: Simulated phenomenon determining the local sensor readings at all nodes. With respect of their distance to the centre of this phenomenon the sensor nodes apply one set of actual sensor readings out of the four listed ones.

of the *fire* producing the highest sensor readings is defined as a circle around the centre point with a radius of 1.5 meters. In this area all sensor readings clearly exceed the defined thresholds. The area between 1.5 to three meters specifies the immediate vicinity of the *fire* centre, where the sensor readings slightly exceed the thresholds and the phenomenon is still visible. The area between three and six meters defines the outer expansion of the phenomenon. In this area the sensor readings are slightly below the thresholds. Finally, all nodes not located in one of these three areas of the phenomenon generate "usual" sensor readings that are clearly below the thresholds.

To trigger changes in sensor readings and node evaluation results, the centre of the *fire* phenomenon deterministically moves within the network boundaries at every minute or six evaluation intervals respectively. In comparison to the spatial expansion covered by the entire sensor network the size of the phenomenon is rather small. Most sensor nodes will therefore generate negative evaluation results per interval whereas only a few nodes may possibly detect the phenomenon. This perfectly fits to the fire detection scenario where upcoming fires usually feature a small size. It is quite obvious that such behaviour does not correspond to a fire in the real world. However, the introduced fire detection scenario is a well descriptive vehicle to exemplify event definition and reliable event detection within a mission-critical context. The described phenomenon is only used to generate deterministic sensor readings and simulation results. The case that

sensor nodes may be damaged or destroyed by such phenomenon is also not considered.

4.6.3 Deviations in sensor readings

This section examines the impacts of deviating sensor readings to the final evaluation result and analyses the effectivity of Majority Voting (MV) and Reactive Majority Voting (RMV) when those are applied as possible countermeasures. Failures in sensing devices attached to the sensor nodes or various influences from the environment, e.g., caused by high humidity or temperature, may lead to uncertain detection results [11]. Due to the fact that the sensors need to be low cost it is further not possible to use highly reliable sensing devices. Hence, sensing devices attached to the nodes in principle possess a certain error of measurement. These errors may cause sensor readings to deviate from the correct ones. In the following scenarios the deviations attached to the sensing devices in the sensor nodes vary from -46 to +46 percent. In dependence of an initial vector, which enables to reproduce the errors in measurements for each simulation run, all nodes apply pseudo randomly chosen deviations. Three separate scenarios regarding only positive, only negative and general deviating measurements were simulated. Each scenario was separately executed in standard mode and in application of Majority Voting (MV) and Reactive Majority Voting (RMV) to finally compare the detection results and the associated overhead of each method.

Total detection accuracy in %			
(Figures B.1 and B.2)	Standard	MV	RMV
Uniform grid deployment	89.819	89.600	**93.626**
Uniform random deployment	87.083	86.510	**91.775**
Random; voting region = 2m	87.083	86.267	**90.525**
Random; voting region = 1m	87.083	86.389	**88.875**

Average of voting messages per node and interval		
(Figure B.3)	MV	RMV
Uniform grid deployment	4.566	**0.734**
Uniform random deployment	4.459	**0.891**
Random; voting region = 2m	3.180	**0.628**
Random; voting region = 1m	1.577	**0.301**

Average of detected events per interval				
(Figures B.4 and B.5)	Reference	Standard	MV	RMV
Uniform grid deployment	4.999	15.189	15.392	**6.686**
Uniform random deployment	5.954	18.870	19.438	**12.898**
Random; voting region = 2m	5.954	18.870	19.629	**14.289**
Random; voting region = 1m	5.954	18.870	19.564	**16.764**

Number of *False positives* in %			
(Figures B.6 and B.7)	Standard	MV	RMV
Uniform grid deployment	10.181	10.384	**3.078**
Uniform random deployment	12.929	13.497	**7.024**
Random; voting region = 2m	12.929	13.676	**8.348**
Random; voting region = 1m	12.929	13.611	**10.810**

Number of intervals with missed phenomenon in %			
	Standard	MV	RMV
Uniform grid deployment	0.0	0.0	0.0
Uniform random deployment	0.0	0.0	0.0
Random; voting region = 2m	0.0	0.0	0.0
Random; voting region = 1m	0.0	0.0	0.0

Table 4.2: Comparison of applying MV and RMV in case of positive deviating sensor readings. This table briefly summarises the results of the diagrams which can be seen in the listed Figures. In both deployment scenarios the RMV enhances the accuracy of detection by about five percent while requiring less than one message per interval and node. In contrast to that, the MV approach behaves nearly equal to the standard detection but requires a huge number of voting messages. For details refer to the following Section and to the diagrams linked in the table.

Positive deviations in sensor readings

Positive deviations in sensor readings usually lead to detection of more events than actually exist in the monitored environment. For monitoring or surveillance applications like fire detection this may result in false alarms. The following evaluates the performance of the sensor network when the sensing devices of the sensor nodes apply only positive deviations of 0 to 46 percent. In this context, the number of *False positives* is usually high, whereas existing phenomena should be reliably detected. In other words, the phenomenon is clearly visible in the detection result since also nodes in the border regions of the phenomenon will most possibly evaluate to TRUE.

A summary of all relevant results of this failure scenario is given in Table 4.2. Whereas using MV makes no impact to the evaluation results compared tot the standard, the RMV increases the detection accuracy in both deployments by four to five percent in comparison to the standard detection. To compare the performance of the approaches, Figure 4.12 displays the total number of detected events (positive results) per interval in comparison to the existing events. The number of detected events is quite high as it was expected. In both deployments, the standard detection and the MV detected nearly 200 percent more events than actually existed. The RMV performs much better but different depending on the deployment scenario. In the uniform grid deployment it generates only 33 percent more positive results whereas it doubles the number of detected events in the uniform random deployments. However, RMV still outperforms the standard and MV detection results. The high total number of detected events originates from the high number of *False positives*, see Figure 4.13.

As it was expected from this high number of detected events, there were no intervals in which all sensor nodes missed the existing phenomenon and hence no *fire* remains undetected. The significant discrepancy in detected events and the number of *False positives* between MV and RMV results from the different voting philosophy. MV always verifies the local detection results. Under consideration of positive deviations in sensor readings, already detected events will most probably remain positive after the voting procedure. Furthermore, formerly negative results from nodes featuring short distances to the phenomenon may be overruled to be positive, too. Besides many originally positive results due to the deviations, this may lead to additional *False positives*. In contrast to that, RMV only votes on detected events. This way the total number of voted events is less or equal to the number of detected events. Since negative evaluation results are not voted, there exists no chance that a formerly negative result is overruled to appear as a positive result as it happens in MV. According to this, RMV at most reduces the number of originally detected events.

It is quite obvious that every fault detection method requires some overhead. In case of voting, this overhead can be represented by the number of required

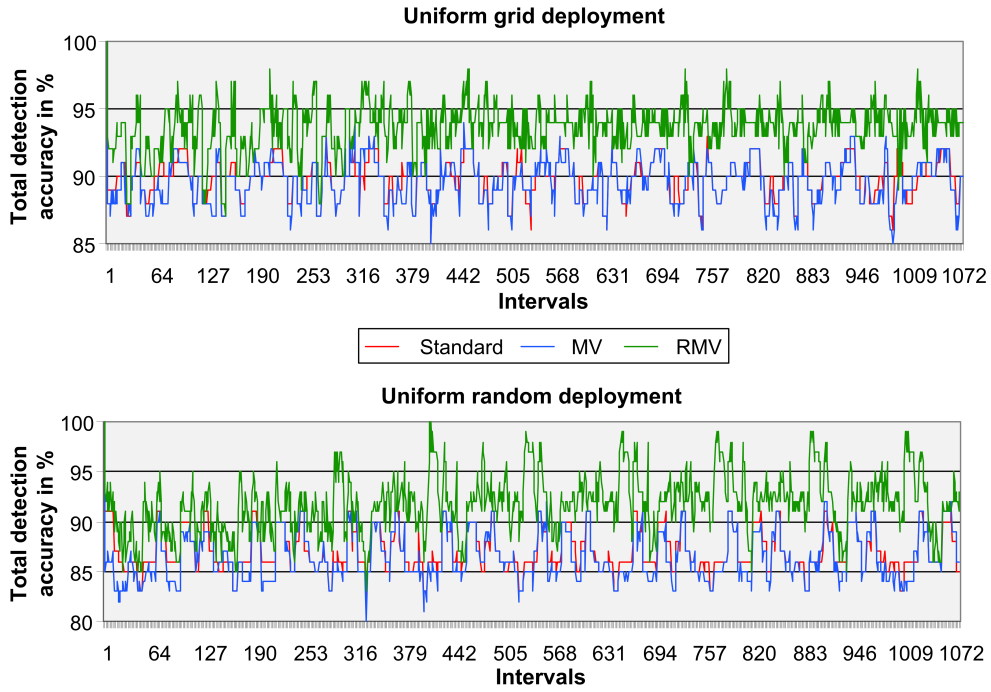


Figure 4.12: Comparison of total detection accuracy when applying MV and RMV in case of positive deviating sensor readings. By reducing the number of *False positives*, the RMV approach enhances the accuracy of detection by about 5%. In contrast to that, the MV approach performs nearly equal to the standard detection.

voting messages. Here, the advantages of the RMV become clearly apparent. It not only increases the detection accuracy but also requires significantly less messages than MV. In average, the RMV required less than one messages per detection interval and node. This reduction in comparison to MV is reached by omitting voting in case of negative detection results. In this scenario RMV reduced the number of voting messages by a factor of five. This factor is expected to be even higher in scenarios with less detected events as it is presented in the next sections.

For further evaluation also the usage of smaller voting regions was tested. This feature is provided by the ESL and allows to fine tune the voting behaviour. With downsizing the voting region the detection performance converges to the standard detection due to the decreasing number of available voters in smaller regions. For this scenario downsizing of the voting region negatively affected the detection performance of RMV. It increases the number of *False positives*, see Figure 4.14, due to the fact that the possibility to overrule wrongly detected events also decreases when less voters are available. Hence, downsizing the voting region decreases the total detection accuracy as well. The application of different

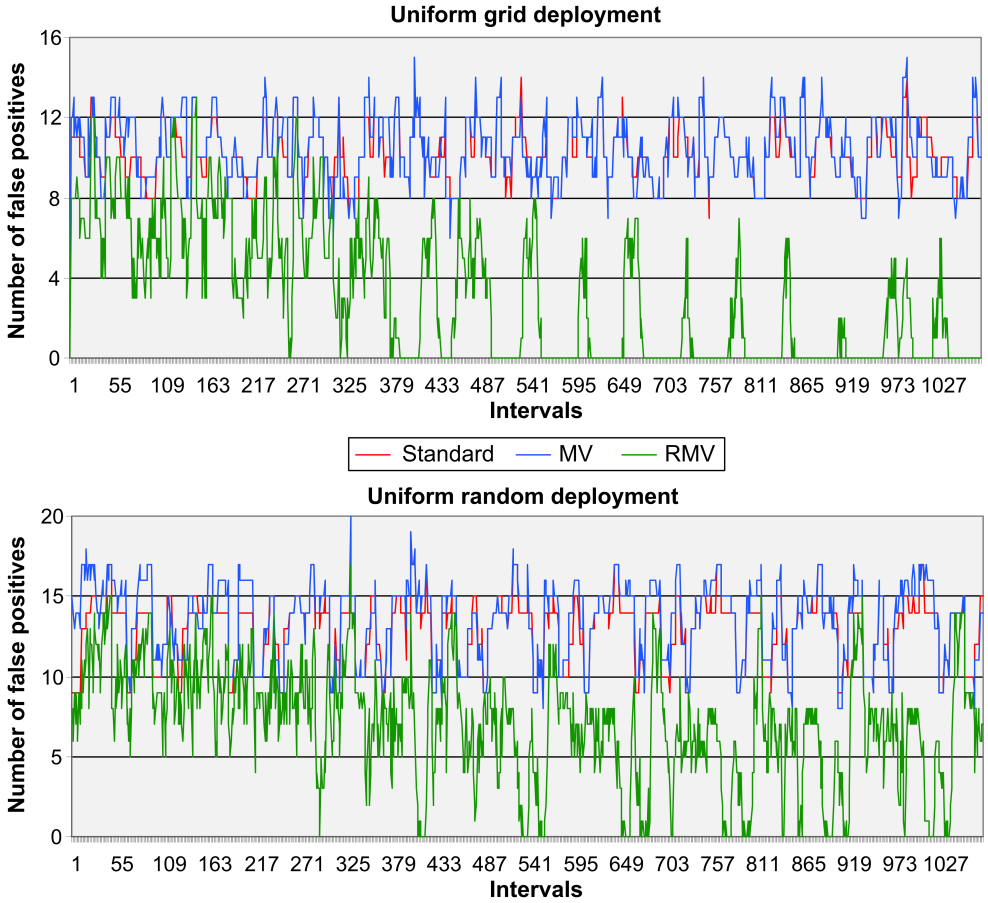


Figure 4.13: Comparison of the number of *False positives* per interval between the standard detection and voting. Here, MV and RMV apply a voting region of 2.5 meters. All approaches gather large numbers of *False positives* while RMV at least reduces these to 30% and 55% compared to the standard detection.

voting regions was tested on uniform random deployments only. In the grid deployment featuring node distances of at least 2.5 meters between all nodes such downsizing is useless since it would merely separate all nodes into different voting regions. In that case the detection results are equal to the results of the standard detection but still requires overhead.

In summary, the usage of RMV applying the original voting region of 2.5 meters provides the best reliability of fire detection in case of positive deviating sensor readings. It enhances the performance in comparison to the standard detection by reducing the number of *False positives* while no *fire* remains undetected. Nevertheless, RMV only requires an acceptable number of voting messages. This is exactly the application that RMV was designed for. It only regards positive detection results and evaluates these by voting. If there is no event at all, a voting as it is done by MV is absolutely unnecessary. Due to the fact that

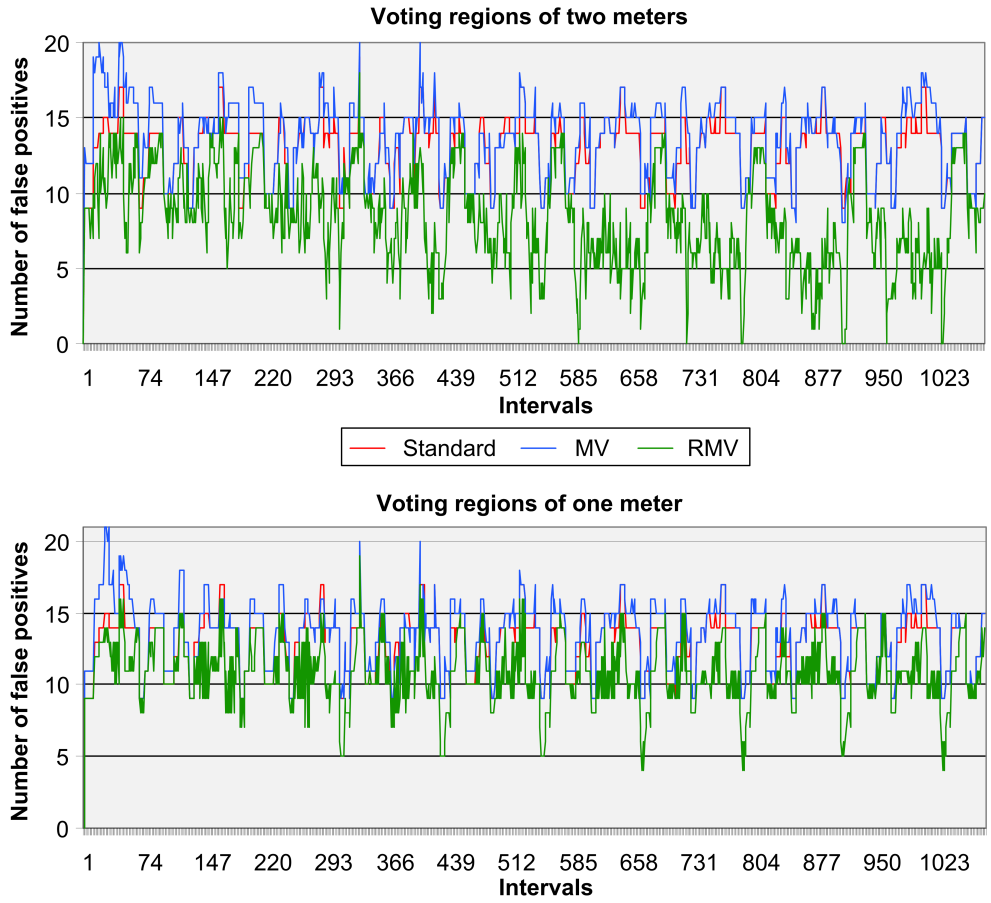


Figure 4.14: Comparison of *False positives* per interval between the standard detection and voting. Here, MV and RMV apply voting regions of 2 meters and 1 meter. With downsizing the voting region the number of available voters decreases. For RMV this increases the number of *False positives* due to the fact that less events are overruled by other devices.

all nodes participating in the voting would announce a negative result as well, it would still gather a negative result.

Total detection accuracy in %			
(Figures B.8 and B.9)	Standard	MV	RMV
Uniform grid deployment	**97.347**	96.892	95.126
Uniform random deployment	**97.152**	97.016	94.601
Random; voting region = 2m	97.152	**97.154**	95.124
Random; voting region = 1m	97.152	**97.157**	96.811

Average of voting messages per node and interval		
(Figure B.10)	MV	RMV
Uniform grid deployment	4.566	0.117
Uniform random deployment	4.459	0.156
Random; voting region = 2m	3.180	0.108
Random; voting region = 1m	1.577	0.043

Average of detected events per interval				
(Figures B.11 and B.12)	Reference	Standard	MV	RMV
Uniform grid deployment	4.994	**2.341**	1.945	0.120
Uniform random deployment	5.954	**3.106**	3.086	0.554
Random; voting region = 2m	5.954	3.106	**3.211**	1.078
Random; voting region = 1m	5.954	3.106	**3.255**	2.765

Number of *False positives* in %			
	Standard	MV	RMV
Uniform grid deployment	0.0	0.0	0.0
Uniform random deployment	0.0	0.003	0.0
Random; voting region = 2m	0.0	0.005	0.0
Random; voting region = 1m	0.0	0.012	0.0

Number of intervals with missed phenomenon in %			
(Figures B.13 and B.14)	Standard	MV	RMV
Uniform grid deployment	**4.453**	12.430	93.506
Uniform random deployment	**5.009**	6.215	75.232
Random; voting region = 2m	**5.009**	5.653	53.661
Random; voting region = 1m	5.009	**4.912**	7.507

Table 4.3: Comparison of applying MV and RMV in case of negative deviating sensor readings. Here, the standard detection performs best because the voting algorithms tend to overrule and respectively negate the inherently few positive results. With downsizing the voting region the detection performances of both voting approaches approximate to the standard detection. As a result, MV closely meets the results of the standard but never justifies the required overhead. For details refer to the following Section and to the diagrams linked in the table.

Negative deviations in sensor readings

The second failure scenario considered the same degrees of deviations but all being negative, i.e., the error of measurement deviates from 0 to -46%. Hence, this failure scenario most probably results in undetected events or completely missed phenomena, which obviously is the worst case scenario for fire detection. The brief summary of this scenario is given in Table 4.3. With regard to the total detection accuracy the standard detection method performs best in comparison to the voting approaches applying a 2.5 meters voting region because these tend to overrule detected events. In consideration of the small number of detected events and the significant amount of undetected phenomena, the high total detection accuracy of all approaches is a misleading report. Due to the fact that the phenomenon to be sensed is very small and its influence on the sensor readings is additionally reduced by negative errors of measurement, this most probably leads to negative evaluation results. With regard to the results of the entire network, which are dominated by negative results due to the small size of the phenomenon, this is most probably correct. For comparison, even in the reference scenario only five to six percent of all nodes signal an event.

The most important issue of this failure scenario are the number of detected events and missed phenomena. The standard detection still detects about the half of all existing events while the voting approaches perform significantly different. Whereas the number of detected events in application of MV is slightly behind the result of the standard detection, RMV performs absolutely bad by missing the existing phenomena in 75% to 93% of all intervals, see diagrams 4.15. RMV negates the inherently few positive detection results. In contrast to that, with MV that also votes on negative results some nodes are overruled to true. Hence, nodes within the phenomenon gathering negative results caused by errors in measurement can be overruled. Therefore it is sufficient to reach a tie in the voting, since a tie is treated as a positive result, too. In other words, if a voting region includes an even number of positive and negative detection results, the voting result is positive. In that case, MV overrules originally detected negative results whereas RMV does not trigger a voting at all. This effect allows MV to partially detect more events than the standard detection. Very rarely this may result in *False positives*. In this failure scenario the number of *False positives* is negligible due to the fact that these occur at most at one node per interval. In comparing the cost-efficiency of voting, the MV performs as usual requiring more than four messages per node and interval. While RMV does not provide good detection results at all, it produces at least only marginal overhead by generating one voting message within nine detection intervals.

To improve the final voting results it may be reasonable to configure smaller voting regions. With downsizing the expansion of the voting region from initial 2.5 meters down to two or one meters, both voting approaches perform better with

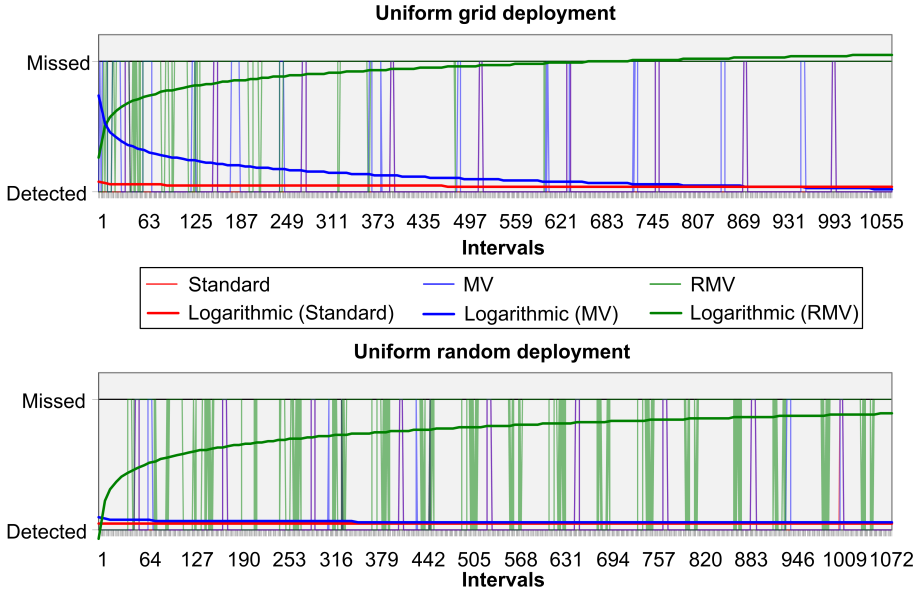


Figure 4.15: Intervals with an undetected (missed) phenomenon. For better visualisation, the overall performance of all detection methods is represented by logarithmic trend curves. These trends represents the median rates of undetected phenomena. The performance of both voting approaches is unacceptable, but RMV performs significantly bad and misses 93 % of existing phenomena.

respect to the number of undetected phenomena. In fact, the results of applied MV are even slightly better than the standard results due to the mentioned effects. However, this marginal increase in detection accuracy still does not justify the number of required voting messages. For RMV the decreased number of available voters increases the detection of events to 46% using a voting region of two meters and 92% when using a voting region of one meter, see diagrams 4.16. Based on that, RMV reverses the mentioned effect of a tie in voting. With RMV voting on positive results only, there have to be more negative votes than positive ones to overrule and respectively negate a detected event. Of course, the probability that detected event are overruled decreases with having less voters available. Nevertheless, the results of RMV are still worse in comparison to the other approaches, which detect about 95% of existing phenomena. Even the only marginal overhead of 0.1 and 0.04 voting messages per node and interval does not save the overall performance of RMV.

This failure scenario clearly announced the drawbacks of RMV, which is rather unsuitable to be used in case of very small phenomena that can be detected by a few or single nodes only. For reliable detection of such phenomena the node density should be increased. Using smaller voting regions further decreased the influences of negative votes and hence, increased the performance of voting. Anyway, in comparison to the detection results of the standard performance both

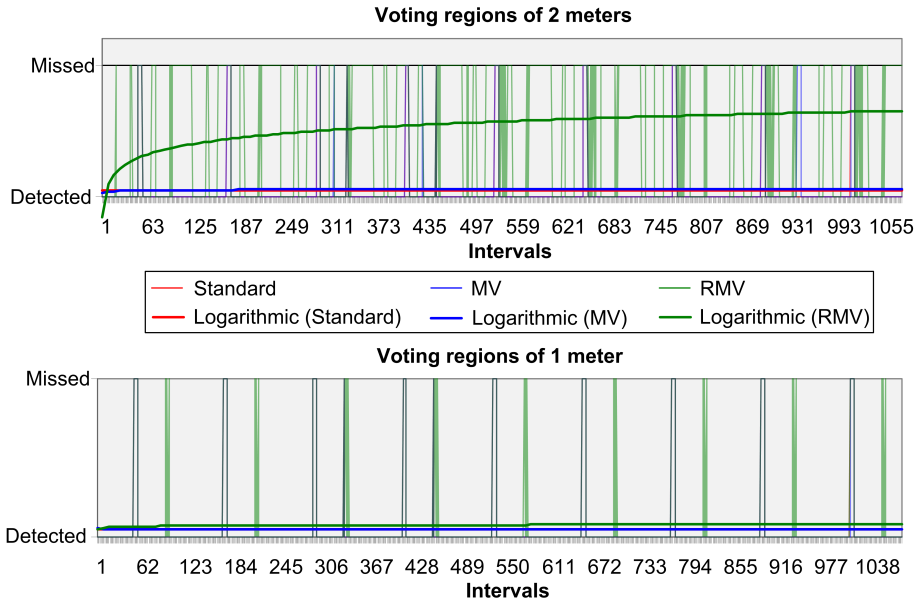


Figure 4.16: Intervals with an undetected (missed) phenomenon. The overall performance of all detection methods is represented by logarithmic trend curves. These trends represents the median rates of undetected phenomena. Downsizing the voting region reduces the number of missed phenomena. All approaches then converge to the standard detection. This is still unacceptable due to the overhead associated to voting.

voting approaches do not justify the required overhead. In view of the detection reliability, voting has to be omitted in scenarios where only a few or single nodes are able to detect the existing phenomenon.

Total detection accuracy in %			
(Figures B.15 and B.16)	Standard	MV	RMV
Uniform grid deployment	93.095	93.120	**95.295**
Uniform random deployment	91.714	92.075	**94.544**
Random; voting region = 2m	91.714	91.729	**94.093**
Random; voting region = 1m	91.714	91.603	**92.971**

Average of voting messages per node and interval		
(Figure B.17)	MV	RMV
Uniform grid deployment	4.566	**0.457**
Uniform random deployment	4.459	**0.534**
Random; voting region = 2m	3.180	**0.376**
Random; voting region = 1m	1.577	**0.177**

Average of detected events per interval				
(Figures B.18 and B.19)	Reference	Standard	MV	RMV
Uniform grid deployment	4.994	9.372	9.043	2.557
Uniform random deployment	5.954	10.943	10.862	3.639
Random; voting region = 2m	5.954	10.943	11.112	**5.564**
Random; voting region = 1m	5.954	10.943	11.186	**8.875**

Number of *False positives* in %			
(Figures B.20 and B.21)	Standard	MV	RMV
Uniform grid deployment	4.376	4.073	**0.515**
Uniform random deployment	4.985	4.918	**0.601**
Random; voting region = 2m	4.985	5.165	**1.114**
Random; voting region = 1m	4.985	5.234	**3.136**

Number of intervals with missed phenomenon in %			
	Standard	MV	RMV
Uniform grid deployment	0.0	0.0	18.275
Uniform random deployment	0.0	0.0	12.059
Random; voting region = 2m	0.0	0.0	0.371
Random; voting region = 1m	0.0	0.0	0.0

Table 4.4: Comparison of applying MV and RMV in case of general deviating sensor readings, i.e., the occurrence of positive and negative deviations. In both deployment scenarios the RMV enhances the accuracy of detection by two to three percent while requiring about 0.5 messages per interval and node. The best detection accuracy of RMV is reached by using a voting region of 2 meters. Again, the performance of MV is similar to the standard detection but requires the overhead of voting. For details refer to the following Section and to the diagrams linked in the table.

General deviations in sensor readings

Finally, a failure scenario applying a mix of positive and negative deviations was simulated. In this scenario, the half of all nodes possess a pseudo-randomly chosen error of measurements between 0 and +46%. Consequently, the other half of all sensor nodes possess errors of measurements between −46% and 0. Such failure scenario is considered to best represent the behaviour of real WSNs. The brief summary of the results gathered by this failure scenario is given in Table 4.4. Due to the simultaneous existence of positive and negative errors of measurements, the detection results differ in dependence of the existing phenomenon and its position. There is no uniform failure behaviour as it was presented in the previous failure scenarios, where either too many or too less events were detected. Hence, it is necessary to compare the detection results of voting to the reference and to the standard detection.

The total detection accuracy of the standard detection and MV perform nearly equal. The RMV enhances the total accuracy of detection by two to three percent. That is again caused by the lower number of detected events due to the fact that some events were overruled. The standard detection almost doubles the number of detected events compared to the reference. With the given phenomenon, the negative errors of measurement affect the total number of detected events less than the positive errors of measurement. In the reference, there exist only five to six percent of positive results, which represent the nodes within the phenomenon. Only these nodes can be affected by negative deviating sensor readings. The results at all other nodes are not affected by negative deviating sensor readings. These nodes still announce a negative detection result. In contrast to that, positive deviating sensor readings may possibly affect the results of 94% of the sensor nodes. According to that, the number of detected events in the standard detection is doubled, see diagrams 4.17.

Of course, the half of detected events in the standard detection represent *False positives*. The application of MV only slightly reduces the number of detected events. It overrules positive results at single nodes in far distance to the phenomenon but also overrules results at nodes near or within the phenomenon, which were affected by negative deviations. In total, this only marginally reduces the number of detected event and *False positives*. RMV combines both behaviours known from the previous failure scenarios. On the one hand, it significantly reduces the number of inadvertently detected events, especially those that are far from the phenomenon. Here only the nodes within the phenomenon remain positive. On the other hand, wrong negative results from nodes within the phenomenon remain negative. In fact, this results in few *False positives* but also misses the existing phenomenon in 12% and 18% of all intervals in dependence of the deployment pattern. Comparing the costs of voting, the RMV reduces the number of voting messages by a factor of nine in comparison to MV. In detail,

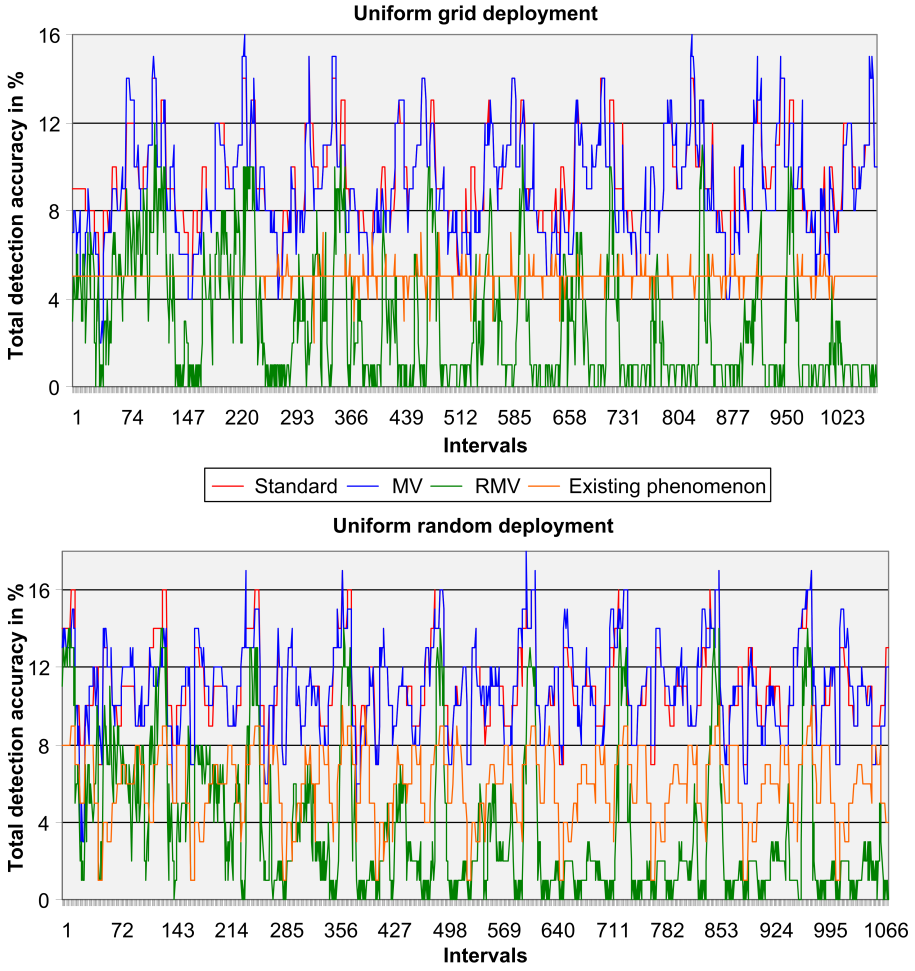


Figure 4.17: Number of detected events per interval in case of general deviating sensor readings. The standard detection and MV almost double the number of detected events. In contrast to that, RMV performs contrary by missing almost half of all existing phenomena.

RMV gets by with about 0.5 or less messages per interval and node.

The performance of RMV strongly depends on the size of the voting region. If the voting region is downsized to two meters, it yet allows to detect 73% of all existing events plus 18.5% of *False positives*. A voting region of one meter is yet too small as the performance converges close to the standard detection then, see diagrams 4.18. A voting region of one meter generates 48% more positive results than the reference scenario. Therefore the marginal decrease in the overall detection accuracy of 0.4%, which results from the increased number of *False positives*, is acceptable. In addition, downsizing the voting region to two meters once again reduces the number of required voting messages by 30%. In contrast to that, even in case of downsizing the voting region the results of MV are again

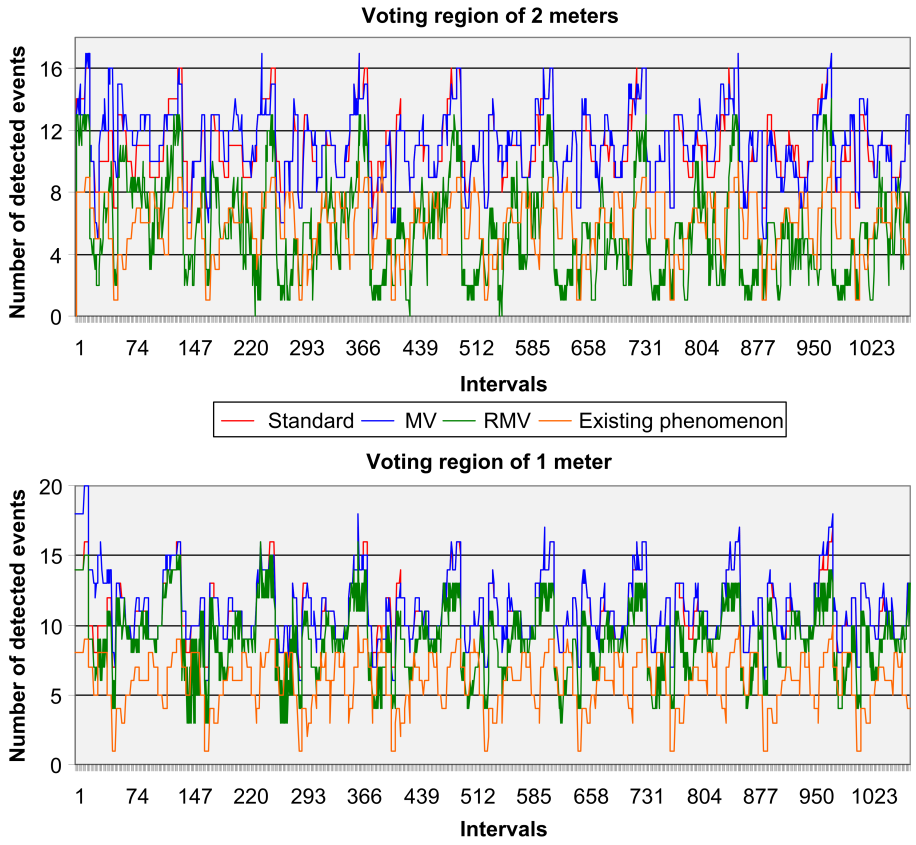


Figure 4.18: Number of detected events per interval when applying different voting regions in case of general deviating sensor readings. In general, downsizing the voting region increases the number of detected events. Here, the RMV achieves by far the best results of all approaches (93% accuracy compared to the reference) by applying a voting region of 2 meters. Choosing a voting region of 1 meter for is already too small for RMV and results in an increased number of *False positives*.

very similar to the results of the standard detection but MV requires a significant overhead for the voting messages. In particular, smaller voting regions increase the number of detected events and *False positives* due to the mentioned effect of even number of votes. Hence, a voting region of two meters is the best setting to provide a sufficient detection accuracy.

For this failure scenario RMV applying a voting region of two meters is by far the best trade off in detection accuracy even if this setting still not always detects the phenomenon. In total, there were four out of 1080 detection intervals in which the phenomenon was not detected. Since these are not consecutive intervals, the phenomenon would be detected with a delay of only ten seconds. With regard to the reliability of the fire detection application, RMV performs best due to

significantly reduced number of *False positives*, which are false fire alarms. In comparison to the standard detection, RMV reduced the number of false alarms by 77%. In contrast to that, MV is unsuitable to enhance the reliability of detection in this scenario. Whereas RMV strongly limits the size of the detected phenomenon, MV tends to enlarge the size of the detected phenomenon. It should rightfully be mentioned that MV was not designed for monitoring scenarios featuring local phenomena only. It is designed to evaluate the correctness of all local detection results when these need to be exactly evaluated, e.g., if small differences in temperature readings have to be detected. It is also designed to detect and revise general sudden deviations in sensor readings. MV is further usually applied at predefined or fixed voting regions where certain restrictions are met, e.g., where a sufficient number of voters is available. It has so far not been tested on unfixed voting regions that introduce an uncertainty in the available voting preconditions. The fact that MV performs rather poor in surveillance applications was also indicated by previous work on voting [34, 64].

4.6.4 Failing sensing capabilities

Low cost production, decreasing energy supply and various influences from the environment may not only lead to errors of measurement in sensor readings. These also cause sensing devices to fail transiently or to get even permanently lost. In that case, usual local event detection based on own sensor readings is limited or cannot be provided further. Of course, this results in a decreased detection accuracy. Collaboration between sensor nodes exchanging missing information is a proper means to keep the functionality of the sensor network and its configured applications alive. In the context of EDTs, this requires to exchange values of EDT nodes. This Section analyses the performance of event detection under random permanent and transient failures of sensing devices. Therefore the results of the standard detection are compared to the detection results in application of ACK-based and lease-based collaboration. Here again, two worst case failure scenarios are simulated. Each all available sensing capability, i.e., the temperature, the carbon monoxide and the smoke sensing devices, will fail on each sensor node over time. Fortunately such extreme failure scenario is far away from real deployments. However, this is necessary to test the given collaboration schemes in worst case scenarios.

In such failure scenario, the total detection accuracy is the most important issue. In contrast to voting, the collaboration schemes are not designed to explicitly enhance the detection of events only. They are rather designed to improve the robustness of the sensor network against failed sensing devices in the sensor nodes and to keep the applications running at all nodes. Therefore, collaboration schemes perform independent from the final detection results. Nevertheless, the detection of events is separately considered to regard the mission critical context.

Total detection accuracy in %				
(Figure B.22)	Standard	Lease=6	ACK	Lease=30
Grid deployment*	66.960	88.760	91.545	-
Grid deployment	59.763	**80.541**	-	-
Random deployment*	63.970	82.917	85.619	84.178
Random deployment	59.763	77.968	-	**79.557**

Average of collaboration messages per node and interval					
(Figure B.23)	Lease=6	ACK	Lease=30	Lease=6 (reliable)	Lease=30 (reliable)
Uniform grid*	0.376	7.721	-	0.584	-
Uniform grid	0.374	-	-	0.561	-
Uniform random*	0.357	7.360	0.079	0.537	0.121
Uniform random	0.356	-	0.078	0.525	0.118

Average of detected events per interval					
(Figure B.24)	Reference	Standard	Lease=6	ACK	Lease=30
Uniform grid*	4.994	3.405	**4.323**	3.402	-
Uniform grid	4.994	3.022	**3.856**	-	-
Uniform random*	6.008	3.747	**5.163**	3.748	4.613
Uniform random	5.954	3.504	**4.858**	-	4.348

Average number of *False positives* per interval				
(Figure B.25)	Standard	Lease=6	ACK	Lease=30
Uniform grid*	0.0	0.292	0.008	-
Uniform grid	0.0	0.259	-	-
Uniform random*	0.0	0.338	0.021	0.228
Uniform random	0.0	0.315	-	0.212

Average number of undetected events				
(Figure B.26)	Standard	Lease=6	ACK	Lease=30
Uniform grid*	1.602	0.963	1.591	-
Uniform grid	1.984	1.398	-	-
Uniform random*	2.261	1.183	2.281	1.622
Uniform random	2.449	1.411	-	1.818

Table 4.5: Comparison of applying the lease-based publish/subscribe and ACK-based collaboration in case of permanently failing sensing capabilities. The lease-based approaches perform best and enhance the standard detection by about 30%. The longer leasing time of 30 intervals, which was only tested on the more realistic random deployments, reduces the number of required messages but the shorter leasing time of six intervals performs better with respect to the detected events. Due to the successively increasing number of necessary collaboration messages in the ACK-based scheme, the simulation process has been killed by the simulation environment. The affected runs are marked with an * and represent the last known system state. For details please refer to the following Section and to the diagrams linked in the table.

Permanently failing sensing capabilities

The first scenario analysed the performance of the introduced detection schemes in case of a successive permanent loss of all sensing capabilities. A successive loss of all sensing features of each node in the entire WSN represents a complete operational breakdown for the application running on the sensor nodes. In the introduced fire detection scenario, the three sensing devices measuring carbon monoxide, temperature and smoke at each sensor node fail within the 1080 simulated time intervals. To simulate a slow decrease of detection performance only one sensing device at one sensor node fails per interval. The occurrence of failures is pseudo-randomly distributed. Of course, this requires to successively adapt (prune) the local EDT at each node when a sensing capability fails. It further triggers the collaboration between neighbouring nodes to gather detection results to substitute the locally missing information, i.e., the value of a certain EDT-node. When all sensing devices at a node failed, this is not equivalent to a crash. In that case, the respective EDT degenerated to a tree evaluating the root node only. The sensor node then runs as a bridge node as introduced in Section 4.3.

A brief summary of the simulation results in case of permanently failing sensing capabilities is presented in Table 4.5. Unfortunately, the simulation environment was unable to finish the runs for the ACK-based collaboration scheme. Due to the successively increasing number of necessary collaboration messages the process executing the ACK-based simulation has been killed by the simulation environment. The affected runs are marked with an * in the table and represent the last known system state. This yet indicates that ACK-based collaboration becomes infeasible with a growing number of failures.

First, the results of all approaches, i.e., until the ACK-based simulation has been aborted, are evaluated. Comparing the average of correct detection results in the entire network both collaboration schemes clearly improved the total detection accuracy. The ACK-based scheme performed slightly better than the lease-based one. This result was expected due to the fact that the ACK-based scheme always gathers the actual detection results at each interval. In contrast to that, the lease-based approach in average reflects changes slower depending on the leasing time, even if the lease is optimised to the behaviour of the phenomenon to be sensed. According to the simulated phenomenon, which moves every six intervals, the lease was also set to six. The lease-based publish/subscribe approach most likely will not outperform the ACK-based variant with regard to the detection accuracy. In fact, the goal of the lease-based detection is to provide a detection accuracy that closely meets the result of the ACK-based scheme, but by that it should significantly reduce the necessary message overhead.

The total detection accuracies of all approaches are compared in Figure 4.19. First, the results at the grid deployment are compared. If 50% of the sensor

nodes were not able to perform their tasks using the standard detection method, the lease-based method was still providing a detection accuracy of 90% and the ACK-based method even generated 94% correct detection results. Even if only 30% of all sensor nodes remain able to gather local results, the accuracy of both collaboration schemes is still at 72%, which is an enhancement of 240% compared to the standard approach.

In the uniform random deployments the performance is only slightly worse than in the grid deployment. In case that 50% of the local detection results are lost in the standard detection, the lease-based and the ACK-based scheme still provide a detection accuracy of 84% and respectively 87%. If 70% of detection results are lost in the standard detection, i.e., only 30 nodes remain functional, the collaboration schemes still generate 67% (lease-based) and 70% (ACK-based) correct detection results. This results in a detection improvement of 225% compared to the standard detection.

Considering only the number of detected events, see Figure 4.20, the lease-based approach compensates the slightly lower performance. In both deployments, it misses only about 19% (0.963 per interval) in average of all existing events in comparison to the reference, which is the faultless case. The standard detection and the ACK-based scheme both miss about 32% (1.602 and 1.591) of existing events. Simultaneously, the lease-based method detects significantly more events, but these also include *False positives*. Of course, published values may trigger events at nodes that are not in the area of the phenomenon. Due to the fact that a lease and therefore also the published values are valid for at most six intervals, the EDT-nodes may hold their values for a longer time than these may actually be correct. The respective sensing devices of the publishing sensor node may also fail during the leasing time. In that case, the other detection schemes may not be able to perform the local evaluation anymore.

With respect to the detected events, the standard detection and the ACK-based approach perform nearly equal because both are directly affected by failed sensing units, which were needed to gather the events. The difference in the total detection accuracy is caused by the detection results at nodes in far distance to the phenomenon. In these areas the sensor nodes exchange only negative values of EDT-nodes. Whereas the standard detection fails to perform a local evaluation at nodes with failed sensing units, here the collaboration schemes enable the nodes to continue their event detection. The simulated phenomenon causes in average five to six percent of events. Due to the fact, that 94% to 95% of all nodes do not detect an event, the publication of negative EDT-node values most likely enables other nodes to gather correct negative results.

The lease-based approach significantly outperforms the ACK-based variant in comparison of the number of collaboration messages, see the diagram in Figure 4.21. It only requires about one message within three detection intervals.

The ACK-based scheme needed to send about 7.5 messages in average at each interval, which in total differs from the lease-based scheme by a factor of 20. This is obviously to much traffic to be simulated. Because of this reason the simulation environment killed the respective runs performing the ACK-based collaboration. It further limits the applicability of the ACK-based scheme with respect to the node density of the network and the amount of data that needs to be exchanged.

Only the lease-based approach and the standard detection have completed their simulation runs. The following compares the respective performances. Again, the lease-based approach enhances the total detection accuracy in the network by an average of 35% in the grid deployment and 30% in the uniform random deployment. It further detected 77% (grid) and 82% (random) of all existing events, which represent a gain of 28% and respectively 39% in comparison to the standard method. This significant increase is achieved by requiring an collaboration overhead of only 0.35 messages per node and interval. This is equivalent to the usage of 35 messages in one detection interval representing ten seconds in lifetime. Even using the reliable mode for the lease-based collaboration, i.e., to explicitly confirm each published value, increases the overhead to only 0.53 messages per interval. Please note, the reliable publishing does not influence the detection accuracy. It triggers an explicit acknowledgement for each received publication.

In the uniform random deployments also a longer leasing time of 30 intervals was tested. In the given scenario, this lease provides a better detection accuracy and of course requires significantly less messages, too. Even the somewhat reduced detection of events (72%) is not the main drawback here. Due to the long leasing time, the gathered values are kept much longer even if those do not completely reflect the actual state. Such long leasing time is only suitable if changes regarding the phenomenon occur very rarely. Hence, keeping a negative value for a long time provides a high possibility that this is the correct value. In addition, gathered information is kept even if the sensing capability of the respective sensor nodes already failed in the meantime. Such long leasing time will perform worse in application scenarios that feature more frequent changes in sensor readings or an equally high number of positive and negative detection results.

With an increasing number of unavailable sensing devices the performance of the standard detection continuously decreases. Both collaboration schemes can significantly extend the time of running the event detection with a high detection accuracy. Even if 50% of all sensor nodes cannot evaluate the EDT by own sensor readings, both collaboration schemes still provide a total detection accuracy of at least 90%. As expected, the ACK-based scheme performs slightly better than the lease-based scheme but requires far more collaboration messages to achieve such results. The number of necessary messages differed by factors of 20 or even more depending on the leasing time that is used. It is an unsolved question whether this

overhead will also cause real applications to fail as it happened in the simulations. However, two remarks need to be emphasised. First, this failure scenario is rather unlikely to occur in real deployments. Second, applications running in such failure scenario possess a certain point in time where the detection accuracy falls below the required minimum in either way, regardless of detection enhancement. A sensor network that features too many failures should be renewed or switched off.

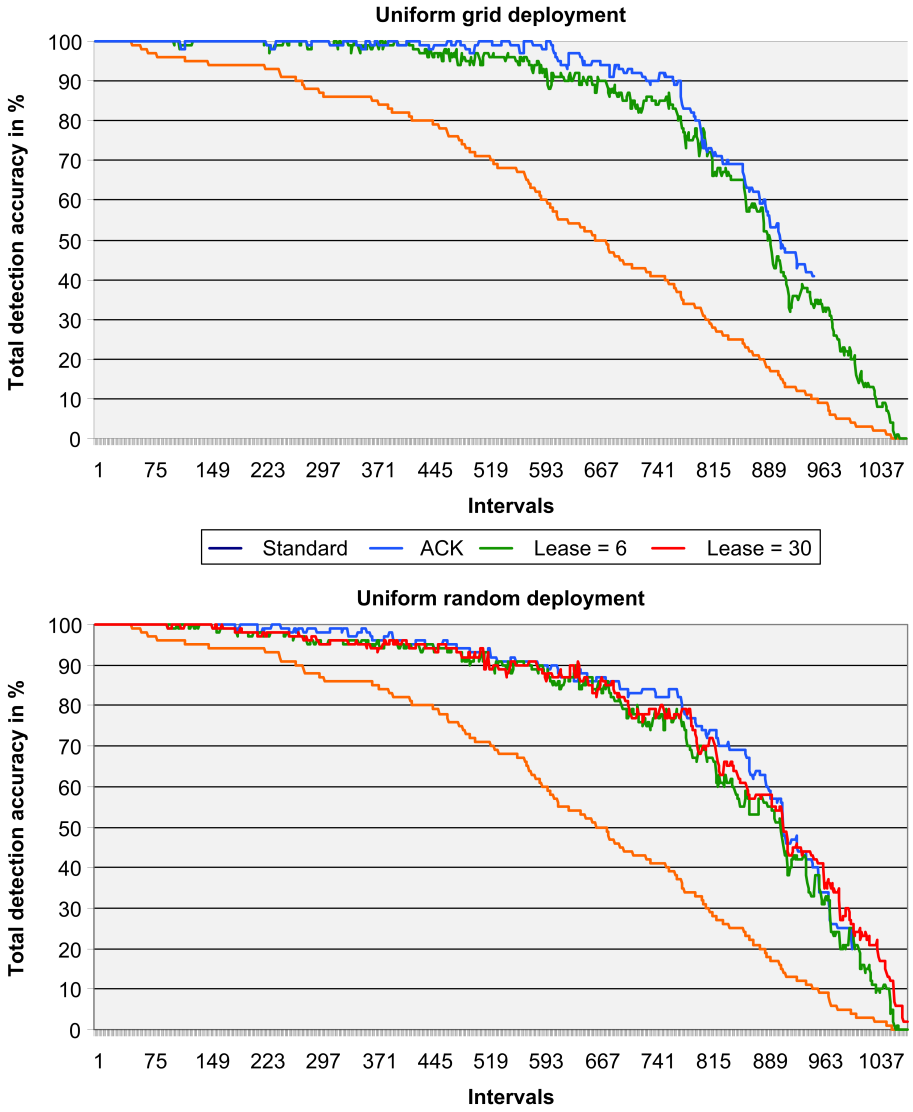


Figure 4.19: Comparison of detection results when applying lease-based and ACK-based collaboration in case of permanent failing sensing capabilities. In both deployments, the lease-based approach enhances the detection accuracy by 30% to 35%. The ACK-based scheme indicated similar or even better performance but has been aborted by the simulation environment due the huge number of messages needed. Moreover, even if only 50% of all nodes are able to generate local detection result in the standard scheme, both collaboration schemes still provide an detection accuracy that is higher than 80%. Please note, the simulation runs of the ACK-based scheme have been aborted by the simulator due to a high number of messages used.

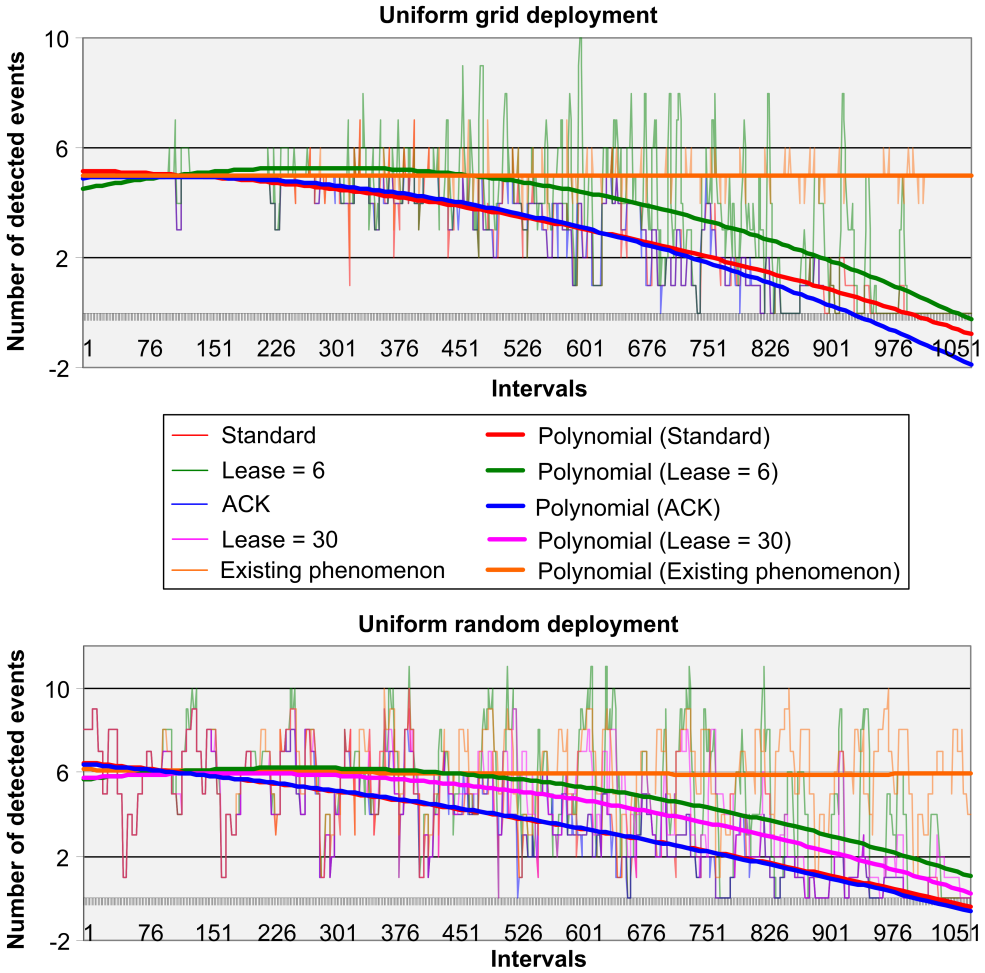


Figure 4.20: Comparison of detected events applying the lease-based and ACK-based collaboration schemes in case of permanent failing sensing capabilities. For better visualisation, the overall performance of all detection methods is represented by polynomial trend curves. These trends represents the median rates of detected *False positives*. The lease-based approach significantly improves the standard detection results by 28% (grid) and 39% (random). In average, it detects 77% (grid) and 82% (random) of all existing events. The ACK-based approach performs well until its abort, which results in a negative trend.

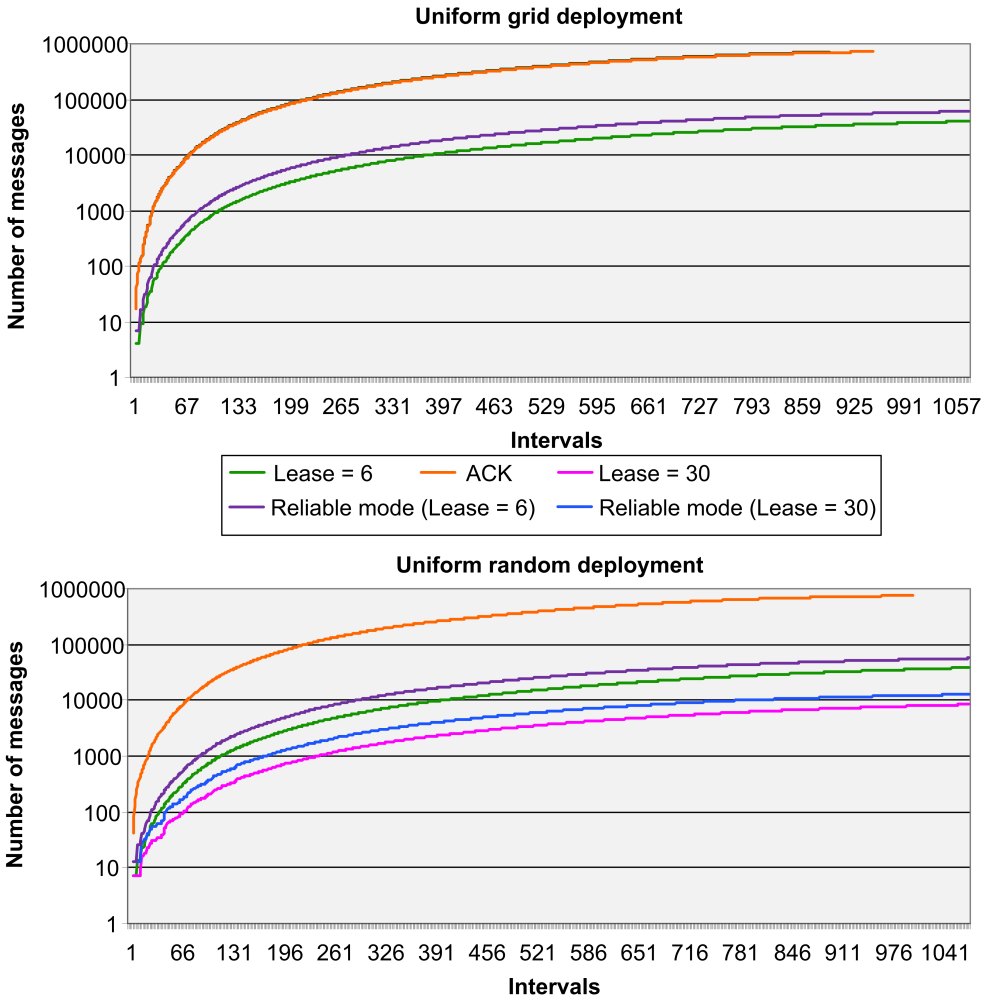


Figure 4.21: Comparison of required messages in the entire network in application of lease-based and ACK-based collaboration in case of permanent failing sensing capabilities. Despite of the significantly enhanced detection performance, the lease-based approach only requires to transmit 0.35 messages per node and interval. Also using the reliable mode in lease-based collaboration, which does not influence the detection results, requires to transmit about one message in two intervals. In contrast to that, the ACK-based approach required in average more than 7 messages per node and interval. This caused the simulation runs to be aborted by the simulator. Please note, the diagrams applied logarithmic scales.

Total detection accuracy in %			
(Figure B.27)	Standard	Lease = 6	ACK
Uniform grid deployment	93.567	99.613	**99.691**
Uniform random deployment	93.567	99.416	**99.563**

Average of collaboration messages per node and interval			
(Figure B.28)	Lease = 6	ACK	Lease = 6 (reliable)
Uniform grid deployment	**0.310**	7.672	0.535
Uniform random deployment	**0.300**	8.089	0.514

Average of detected events per interval				
(Figure B.28)	Standard	Lease = 6	ACK	Reference
Uniform grid deployment	4.643	**4.925**	4.654	4.994
Uniform random deployment	5.6	**6.057**	5.591	5.954

Average number of *False positives* per interval			
(Figure B.30)	Standard	Lease = 6	ACK
Uniform grid deployment	0.0	0.163	0.039
Uniform random deployment	0.0	0.278	0.069

Average number of undetected events per interval			
(Figure B.31)	Standard	Lease = 6	ACK
Uniform grid deployment	0.351	0.233	0.379
Uniform random deployment	0.354	0.175	0.431

Table 4.6: Comparison of applying the lease-based publish/subscribe and ACK-based collaboration in case of transiently failing sensing capabilities. Both collaboration methods perform excellent and feature a detection accuracy of nearly 100% but the lease-based approach required 25 to 27 times less messages. In comparison to the reference, the lease-based detection further features a deviation of only 1.5% in the detection of events. For details refer to the following Section and to the diagrams linked in the table.

4.6.5 Transiently failing sensing capabilities

After evaluation of collaboration in case of a successive permanent loss of all sensing capabilities this scenario considers transient failures in sensing devices. Transient failures cause the sensing devices to fail only for a certain number of intervals before those sensor readings are available again, e.g., in case of intermittent failures by heavy noise or required recalibration. This stresses the designed support of pruning and refreshing the EDTs. Similar to the simulation of permanent failures, at most one sensing device of one sensor node fails per interval.

The occurrence and the duration of failures is pseudo-randomly distributed. Table 4.6 represents the results of simulating transient failures in sensing devices at a glance.

In view of the detection accuracy, displayed in Figure 4.22, both collaboration methods perform excellent. Whereas the standard detection provides an accuracy of 93.5%, both collaboration methods reach a detection accuracy of nearly 100%. That is an outstanding result given that the network temporarily provides only 85% of functional nodes that are still able to provide detection results based on own sensor readings. The comparison of the number of detected events to the respective number in the reference also confirms the pretty good results. The lease-based approach features a deviation of only 1.5% whereas both other methods deviate by about 7%. Due to the leasing time that possibly causes the EDT-nodes to keep positive values even if these are not up to date, the lease-based detection introduces a marginal number of *False positives*. The ACK-based scheme updates the EDT-node values more frequently, which results in less *False positives*. Nevertheless, both methods still provide the possibility that published values from nodes inside the phenomenon may trigger an event at nodes located near the phenomenon. In addition, only less than one event in two intervals was missed, see diagrams 4.23. Since the phenomenon causes about five events per interval in the reference, the remaining detected events by far do not lead to an undetected phenomenon. Finally, even in this scenario the nodes in far distance to the event, that exchange primarily negative EDT-node values, make the big difference in the total detection accuracy.

As it was expected, the lease-based approach also outperformed the ACK-based approach with regard to the required collaboration messages. The lease-based approach requires to only transmit one message per node within three intervals. In contrast to that, the number of ACK-based collaboration needs at least a factor of 25 more messages. To conclude, both methods provide an excellent detection accuracy but only the cost of the lease-based approach is acceptable. In case of transient failures it is suitable to collect EDT-node values from neighbouring sensor nodes until the own sensing device is functional again. This introduces only few *False positives* and undetected events, which are still acceptable.

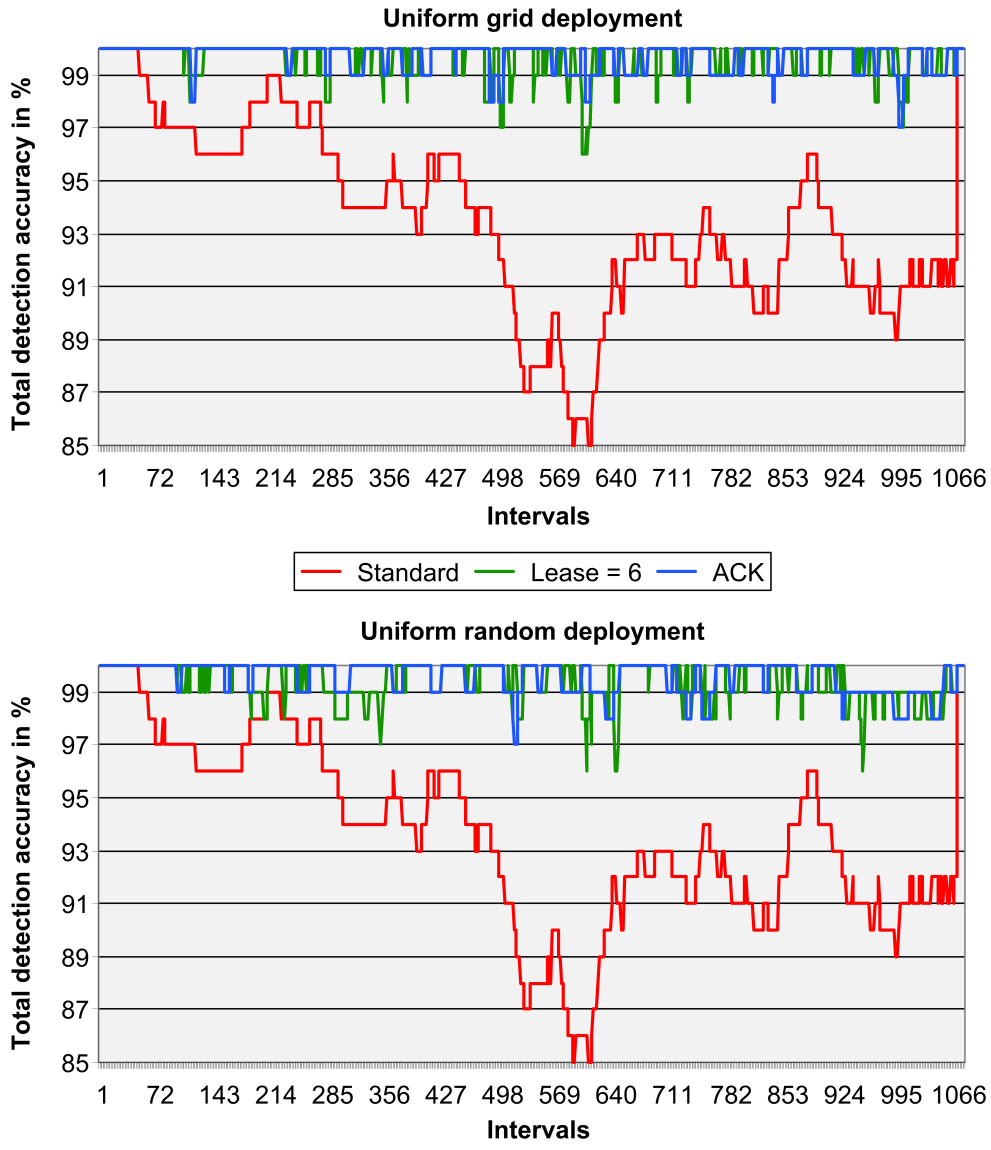


Figure 4.22: Comparison of detection results when applying lease-based and ACK-based collaboration in case of transiently failing sensing capabilities. Both collaboration methods perform excellent and feature a detection accuracy of nearly 100% that in average enhances the standard detection by 6%.

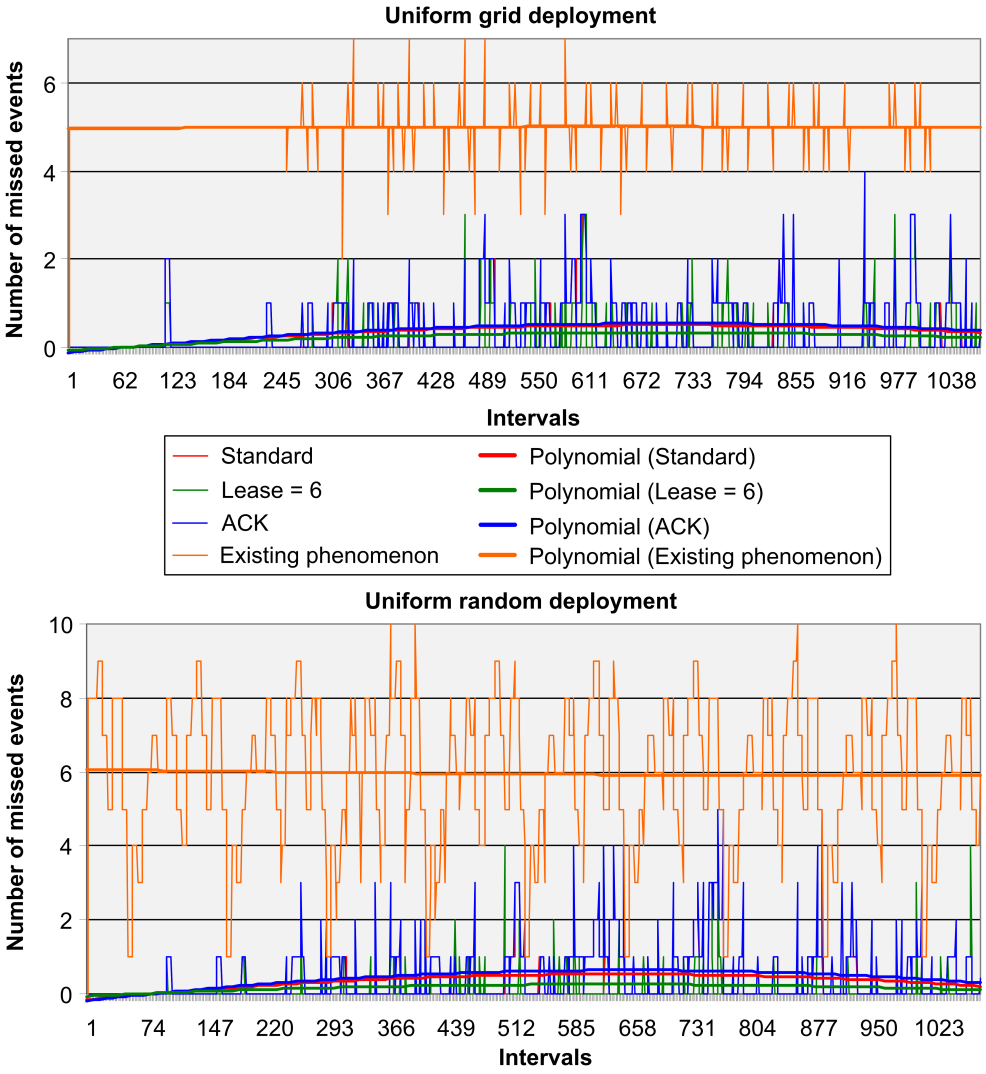


Figure 4.23: Comparison of detected *False positives*. The number of undetected events is rather low. The polynomial trend curves show a similar number of undetected event for the standard detection and the ACK-based scheme. The rate of undetected events is even lower in the lease-based scheme. However, all detection method have not missed a phenomenon.

Total detection accuracy in %						
(Figure B.32)	Standard	MV	RMV	Lease=6	ACK	RMV+Lease
	85.688	86.931	88.272	91.313	91.684	**93.250**
Average of messages per node and interval						
(Figure B.33)		MV	RMV	Lease=6	ACK	RMV+Lease
		4.238	0.479	**0.300**	8.089	0.867
Total number of detected events (Figures B.34 and B.35)						
Reference	Standard	MV	RMV	Lease=6	ACK	RMV+Lease
5.954	10.361	10.405	3.527	11.220	10.350	**8.285**
Total number of *False positives* per interval						
(Figure B.36)	Standard	MV	RMV	Lease=6	ACK	RMV+Lease
	4.419	4.505	**0.986**	5.276	4.412	2.859
Number of undetected events						
	Standard	MV	RMV	Lease=6	ACK	RMV+Lease
Per interval	0.007	0.049	2.541	**0.005**	0.011	0.526
In total	8	53	2741	**5**	12	567
In %	0,125	0,825	42,668	**0,078**	0,187	8,826
Intervals with missed existing phenomena						
	Standard	MV	RMV	Lease=6	ACK	RMV+Lease
In total	0	0	44	0	0	4
In %	0	0	4.078	0	0	0.371

Table 4.7: Comparison of all introduced detection methods in case of general deviating sensor readings and transiently failing sensing capabilities at a uniform random deployment. Each standalone method more or less enhances the detection results in comparison to the standard detection. However, the combination of RMV with the lease-based collaboration scheme provides the best results while requiring an acceptable overhead of less than one message per interval. This already includes all voting and collaboration messages. For details refer to the following Section and to the diagrams linked in the table.

4.6.6 Simultaneous occurrence of deviations and transient failures

To finally compare all presented approaches, those are applied to a WSN that is exposed to the simultaneous occurrence of generally deviating sensor readings and transiently failing sensing devices. This was only simulated on uniform random deployments to gather more realistic detection results. An overview of respective simulation results is given in Table 4.7. Since two different failure classes are

applied simultaneously, each detection method has to be analysed independently. Each method is originally designed to overcome one of both failure classes only. Hence, the detection behaviour differs partially.

As a basis, all introduced detection approaches are applied and compared to the standard detection method, see the diagrams in Figure 4.24. The results of all standalone methods are similar to the respective results in the previous failure scenarios. The standard detection reaches a detection accuracy of 85%. The voting approaches only slightly improved the standard results by about one percent using MV and about three percent using RMV. In this scenario the failing sensing capabilities are the more critical aspect. Hence, the application of the collaboration schemes both increased the detection accuracy by about six percent.

As known from the failure scenario applying general deviations, the standard detection generates a high number of events, which in total are slightly reduced by the failing sensing devices. By that, it detects almost all events and misses no phenomenon at all. Just like in the previous failure scenarios, failed sensing devices do not allow the standard detection to gather any result at some nodes. Hence, the reduced total detection accuracy is here caused by nodes with failed sensing devices and by *False positives*, which result from deviating sensor readings.

In comparison of the voting schemes, MV generates a high number of positive results with a high number of *False positives* caused by deviating sensor readings. As a result, it detects almost all existing events and does not miss an existing phenomenon. In contrast to the standard detection, MV can additionally gather results at nodes with failed sensing devices. Especially at nodes in far distance to the phenomenon, it simply collects the negative results of the neighbouring nodes and sets a negative result at the initiating node, too. This causes a small increase of the total detection accuracy in comparison to the standard scheme.

RMV presents a contrary behaviour. As known from previous failure scenarios, it overrules and thereby significantly reduces the number of detected events. Of course, this methods provides the lowest number of *False positives* but also misses a large number of events. That also causes a high number of intervals with missed phenomena. A phenomenon is missed if no event is detected in the respective interval. This happens in four percent of all intervals, in which the other detection methods detected one or two events, which were overruled here. With regard to the total detection accuracy, the number of detected or undetected events represent only five to six percent of all detection results. Hence, the low number of *False positives* would rather provide a significant gain in detection accuracy, but RMV only triggers a voting in case of a detected event. Consequently, it does not allow to trigger a voting at nodes with failed sensing devices, which in turn reduces the detection accuracy. Finally, RMV performs

better than the standard detection and MV but does not significantly increase the detection accuracy.

The collaboration schemes both well increased the overall detection accuracy. Despite the deviations in sensor readings, both methods perform nearly equal to the failure scenario applying transiently failing sensing devices only. With regard to the number of detected events, *False positives* and undetected events the results closely meet the standard detection results. In addition, both collaboration schemes of course allow to support sensor nodes with failed sensing devices by EDT-node values from neighbouring nodes, which enables them to gather detection results as well. The high number of detected events ensures that no existing phenomenon is missed.

Just like in previous failure scenarios, the lease-based approach keeps gathered values from neighbouring nodes for the subscribed leasing time, even if these values do not perfectly reflect the current readings due to possibly failed sensing devices. By that, not existing events may still be announced based on previously published values whereas new events are triggered by a moved phenomenon. Hence, the lease-based approach detected even more events than all other approaches and thereby provides the lowest rate of undetected events. Of course, it does not miss an existing phenomenon but also features the highest number of *False positives*. In the end, also in this failure scenario the total detection accuracy of the lease-based approach is slightly worse than the total detection accuracy of the ACK-based variant.

The other aspect to consider when comparing the detection performance is the overhead required to reach a certain detection accuracy. Here again, RMV and the lease-based detection outperform their respective counterparts. With respect to the cost-efficiency, the lease-based approach even outperforms the RMV. It yet requires less messages than RMV and additionally provides a well improved detection accuracy.

Finally, a combination of RMV and lease-based collaboration was tested. It turned out that both detection methods well completed each other. On the one hand, the lease-based collaboration clearly increased the detection rate of event at nodes that possess failed sensing devices. On the other hand, RMV significantly overrules a high number of *False positives* caused by the lease-based collaboration. Nevertheless, this detection method still missed an existing phenomenon in four out of 1080 detection intervals, which is a fault rate of 0.37%. Since these are not consecutive intervals, this introduces a delay of ten seconds in detection of the phenomenon. In addition, this detection variant produces an acceptable overhead of less than one message per interval. This already includes all voting and collaboration messages. This is slightly more than the sum of both standalone test runs. The increase in overhead is caused by the increased number of detected events, which trigger a voting of course. In summary, a combination of RMV

and the lease-based publish/subscribe scheme provides a detection accuracy of 93%, which is an increase of 8.8% in comparison to the standard detection. One may expect the combinations of RMV and ACK-based collaboration or MV and lease-based collaboration to generate the best results. Even if that is true, the expected overhead associated to MV and the ACK-based scheme should rather prohibit such combinations since it does not justify the little gain in accuracy. The best trade off between detection accuracy and message overhead for this failure scenario is given by the combination of RMV and lease-based collaboration.

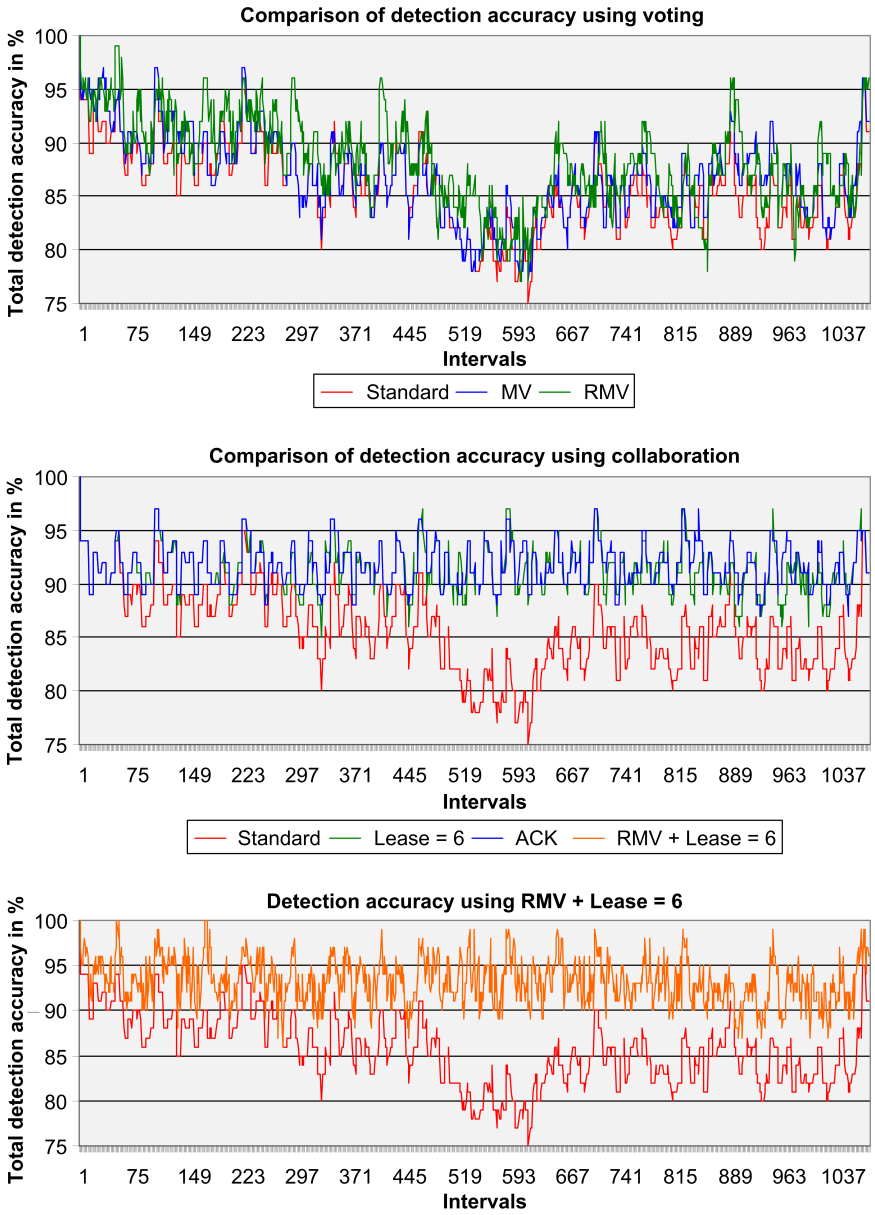


Figure 4.24: Comparison of detection accuracy of all introduced detection methods in case of general deviations and transiently failing sensing capabilities. The collaboration methods clearly perform better than the standard detection and the voting approaches. A combination of RMV and lease-based detection provided the best results with a total detection accuracy of 93%.

	Detection accuracy in %			Message/Interval	
	Standard	RMV	Lease=6	RMV	Lease=6
Positive deviations	87,083	91,775	–	0,891	–
Gain in %		**5,388**			
Negative deviations	97,152	96,811	–	0,043	–
Gain in %		**-0,351**			
General deviations	91,714	94,544	–	0,534	–
Gain in %		**3,086**			
Permanently failing units	59,763	–	77,968	–	0,356
Gain in %			**30,462**		
Transiently failing units	93,567	–	99,416	–	0,3
Gain in %			**6,251**		
Deviations + Transient	85,688	88,272	91,313	0,479	0,3
Gain in %		**3,016**	**6,565**		

Table 4.8: Summary of all total detection accuracies with regard to RMV and lease-based publish/subscribe. In addition, the gain in detection accuracy in comparison to the respective result of the standard detection is presented.

4.6.7 Lessons learnt from simulations

The presented simulation results strongly indicate a need for highly customisable configuration means to fine tune the fault tolerant behaviour to the application scenario, especially with regard to the cost-efficiency of these means. Compared to the improved detection accuracy, the overhead associated with voting and collaboration is worth to be spent. Nevertheless, these methods need to be fine-tuned to achieve a sufficient cost-efficiency. The ESL provides means to customise parameters like the voting region and the leasing time for collaboration. The event detection concept based on ESL and EDT further significantly improved the cost-efficiency of former available MV and ACK-based collaboration by introducing RMV and lease-based publish/subscribe. A summary of all simulation results with regard to these approaches can be found in Table 4.8.

Voting is a proper means to cope with deviating sensor readings. In general, RMV features similar or even better results than MV while it significantly reduces the required number of voting messages. For event-based monitoring scenarios, such as the fire detection application, it is sufficient to concentrate on the results of those nodes reporting an event. The results at all other nodes are of less interest for the task of surveillance. RMV exactly meets theses objectives and intends to isolate the nodes that most probably correctly detected a phenomenon. In addition, that limits the necessary voting overhead to a minimum of required voting messages. However, a sufficient performance depends on the size of the applied voting region, which in turn highly depends on the density of nodes and on the expected size of the phenomenon to be sensed. A high density of nodes

enables to downsize the voting region to such extend that a suitable average number of voters is available. The customisable voting region allows to fine tune the voting procedure to a certain extend but the usability of this parameter is limited, too. If the voting region is chosen too small the sensor nodes may not share the respective voting regions and hence, other voters may not be available. In that case, the results in application of voting converge to those gathered by the standard detection but require a message overhead for the voting request. In the presented node deployments, voting regions of 2.5 meters for the grid deployment and two meters for the random uniform deployments provided the best detection accuracy. Furthermore the voting region should not be larger than the size of the phenomenon. The simulation results indicate a proper average voting region to be smaller than the expansion of the phenomenon. Aiming at a proper ease of use for configuration of voting by non professional users, these must be provided with restrictions indicating a proper size of the voting region. Therefore the following principles apply:

1. The minimum size of the voting region is the mean distance between neighbouring sensor nodes, which is determined by the density of the sensor network, and transmission technology.

2. The maximum size of the voting region is the estimated size of the phenomenon.

If one or both restrictions cannot be guaranteed, the application of voting has to be omitted for cost-efficiency. In that case, there would either be no other device in the voting region (voting region is smaller than minimum) or the event is most possibly overruled by nodes outside the phenomenon (voting region is larger than maximum) and hence, the existing phenomenon remains undetected. To summarise, customised RMV offers proper means to the user to enhance the reliability of detection in event-based surveillance applications. However, the user is responsible for fine-tuning the voting region to achieve a sufficient performance.

Collaboration can significantly enhance the robustness of a sensor network. It keeps on running its applications with a high detection accuracy even in case of failed sensing devices. The presented ACK-based collaboration scheme of course provides the best detection accuracy since it refreshes the actual values of EDT-nodes after each interval but simultaneously requires a huge number of collaboration messages. It was shown that the cost-efficiency of the lease-based approach is very high and reduces the number of collaboration messages by factors of 20 or higher. The lease-based publish/subscribe collaboration scheme can be configured to such extent that it is able to achieve detection accuracies that closely meet those of the ACK-based scheme by choosing a proper leasing time. This leasing time primarily depends on the behaviour of the phenomenon to be sensed. The leasing time ideally is less or equal to the mean time of exposure to

the monitored phenomenon. In the simulations, this was a leasing time of one minute (six intervals). In addition, changes that influence the event detection have also to be considered. Such changes are caused by failed sensing devices and crashed or moved sensor nodes. By that, publisher nodes may get lost or become unable to publish further EDT-node values. Therefore, the estimated mean time to failure has to be regarded, too. For configuration of a proper leasing time, the user has to obey both of the following restrictions:

1. The maximum leasing time is less or equal to the mean time of exposure to the phenomenon to be sensed.

2. The maximum leasing time is less than the mean time to failure.

Of course, the upper bound of a leasing time is one interval. In that case, the performance of the lease-based approach converges to the ACK-based variant. Finally, it is not quite clear whether MV and ACK-based collaboration are generally executable on sensor networks with high node densities and high failure rates. In that case, the sensor network may be unable to manage the amount of traffic associated to those fault tolerant methods. Even if this was possible, it at least significantly stresses the already scarce energy resources and reduces the throughput of the wireless network. Of course, there are dozens of possible test deployments to further stress and analyse the performance of RMV and lease-based publish/subscribe under different conditions regarding varying phenomena, node density, node deployments, unreliable links etc. This is considered to be future work.

Chapter 5

Indicating the Significance of Data Readings

This Chapter presents how to examine behavioural trends in sensor readings to indicate the significance of current measurements beyond the scope of Boolean event decision. Current research of Quality of Information (QoI) has already been introduced in Chapter 2. This chapter first exposes the research objectives before Section 5.1 establishes the variance as the basic math to evaluate the amplitude of actual sensed measurements. Finally, an indicator for the event significance i_S is determined. To illustrate its effectiveness, Section 5.2 presents how i_S affects the quality of detection when applied to the fire detection scenario. Finally, the conclusions examine this approach and discuss future work and further potential application areas.

From the application's point of view, a suitable approach must be independent from both, the types of sensor readings and the applications these are used for. More precisely, such an approach must set objectives to get rid of the necessity to consider application and deployment constraints and hence, it would also be applicable to future and even unknown event definitions. According to that, a proper approach must further be:

- Applicable on every kind of sensor reading.
- Independent from the unit of measurement.
- Efficient in processing and storage.
- Ideally automatically executable without help of the user.
- Useable with and without thresholds.

Especially the last point is a matter of concern, since also novel event detection techniques beyond the scope of thresholds are of interest. Self-learning techniques may enable sensors to determine which sensor reading is probably important and which is not. The current test results indicate a sound basis yet, but these must be evaluated in further simulations and real deployments of course.

5.1 Mathematical background

To determine the significance of sensed measurements requires to contrast actual readings with expected ones learnt from previous trend. It is proposed to apply a maximum likelihood estimate to determine the variance of previous readings σ_m, see Equation (5.1). The variance indicates the range of values where the next reading is most likely in. Since the variance originally requires to use all previous measurements for calculation again and again, it is unsuitable for sensor networks due to the calculation and memory effort. Hence, the standard calculation has been adapted to sensor needs by applying the parallel axis theorem and a customised sliding window derivative.

$$\sigma_m = \sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2};\ \overline{x} = \frac{1}{n}\sum_{i=1}^{n}(x_i) \tag{5.1}$$

The parallel axis theorem allows to process consecutive sensor readings without the need to have all previous values available. Instead, only the sum of measurements and the sum of measurements squares is refreshed and stored, see Equation (5.2). Due to a broad range of applications, the sensing interval usually differs from minutes down to a few milliseconds. Hence, the amount of readings can rapidly increase and result in huge sums used for calculating the variance with the parallel axis theorem. To cope with that, only a number of previous readings specified by an adaptable sliding window s are included. The sliding window provides two benefits. It allows to influence the size of the sums as well as to properly adapt the number of considered measurements to the application. For example, fire detection system are usually not interested in sensed readings of past days, whereas the readings of the last ten minutes may be important for comparison and evaluation.

$$\sigma_m = \sqrt{(\frac{1}{n}\sum_{i=1}^{n}x_i^2) - \overline{x}^2} = \sqrt{(\frac{1}{n}\sum_{i=1}^{n}x_i^2) - (\frac{1}{n}\sum_{i=1}^{n}x_i)^2} \tag{5.2}$$

Applying the parallel axis theorem allows to get rid of stored measurements indeed, but is in principle unsuitable for the sliding window method, which requires these measurements. This approach combines both methods in σ_s by estimating the measurements within the sliding window. The important values

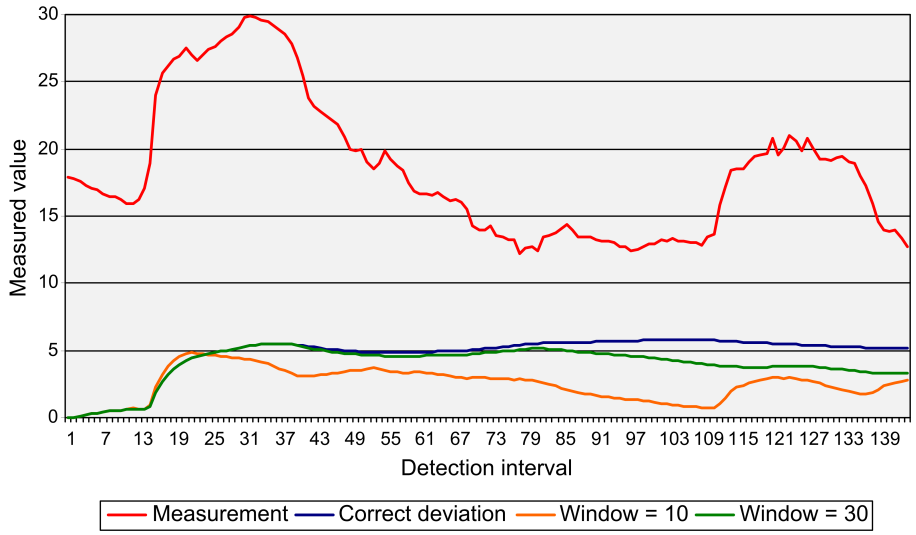


Figure 5.1: Effect of the size of the sliding window to the variance parameter at a series of temperature measurements.

of the parallel axis theorem are the sum of measurements and the sum of measurement squares, as mentioned. Originally, these sums are updated at every sensing interval by processing on the next sensor reading. To apply the sliding window method, the current measurement is added to the sums whereas the expected values are subtracted. The expected values are given by the average of the previous window, see Equation (5.3).

$$\sigma_s = \sqrt{\frac{1}{s}\Big(\sum_{i=n-s-1}^{n-1} x_i^2 + x_n^2 - \overline{x^2}\Big) - \Big(\frac{1}{s}\big(\sum_{i=n-s-1}^{n-1} x_i + x_n - \overline{x}\big)\Big)^2} \tag{5.3}$$

To make sure the sliding window estimation works properly, the algorithm was tested on different series of measurements with changing window sizes. To provide a reference, Figure 5.1 depicts results of applying the estimation approach at a series of temperature measurements. It turned out that the estimated sliding window works fine except for an expected but unavoidable short delay. Due to the estimation of the expected value, bigger changes in measurements completely influence the expected value not before the next sensing interval. However, it also shows the sliding window suitably adapts the variance to recent sensor readings, which allows a proper assessment of the monitored context.

The variance of previous readings (within the sliding window) provides the basis to give a statement about actual readings. It enables to decide whether actual readings meet expected parameters or not. It further allows to classify how far new readings deviate from the expected scope. Therefore the system determines the event significance indicator i_S, which states by what multiple the

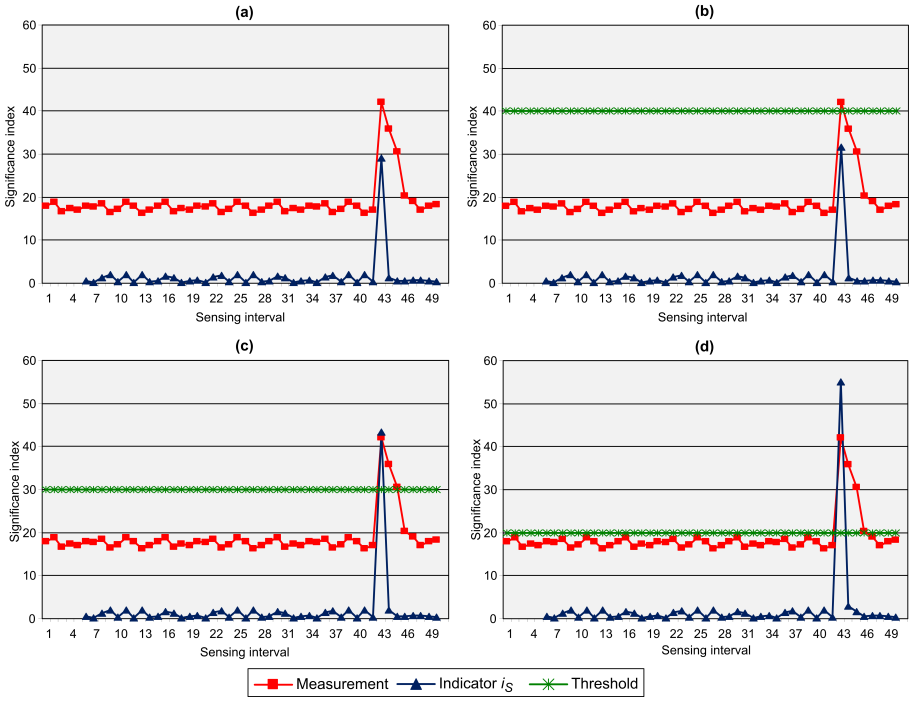


Figure 5.2: Determination of the i_S applied at the same series of temperature measurements with different thresholds. Please note, test case (a) defines no threshold but still allows to clearly detect the sudden temperature increase.

actual reading diverges from the variance. Due to the parameters learnt from the variance of the measurements within the sliding window, the i_S allows to automatically detect significant deviations and trigger proper evaluation without the need to define fixed thresholds for detection. That way, it is expected to enable sensors to use equal event definitions in different environments. For example, fixed thresholds may be unsuitable to be used in the introduced fire fighting system. Whereas a temperature limit of 80 centigrade may be suitable for detecting fires at home, but this limit may be too high to detect fires in buildings, where the ambient temperature is much lower. Hence, the fixed limit is either exceeded too late or never reached in worst case, which possibly results in undetected fires. Similar conditions are given at [22], where acoustic sensors are used for target detection and movement approximation in military application. The applied detection scheme may work well under the given deployment constraints, but must be completely reconfigured to be used in environments with other noise levels.

Besides pure event detection based on i_S, also predefined thresholds are considered to further affect the calculation of i_S. Thresholds usually separate the uncritical range of measurements from the important or critical one. Here it is proposed to double the weight of the critical range in the i_S to regard this importance. It makes indeed a difference whether equal changes in sensed readings

clearly exceed the given thresholds or not. In other words, a temperature increase of 25 Kelvin exceeding the given threshold by 20 Kelvin is more important than the same increase exceeding the threshold by 5 Kelvin only. That especially holds true, if events are evaluated in terms of a mission. This temperature increase of 25 Kelvin was simulated in an example scenario shown in Figure 5.2. Here, the same sequence of temperature readings are used to determine the i_S without a threshold, see diagram (a), as well as with different thresholds (b)-(d). These clearly show that the i_S approach can detect significant changes in sensor readings. Moreover, the i_S improves the detection of the temperature increase with a higher amplitude than the readings do. If thresholds are additionally taken into account, the i_S even grows in proportion to the exceedance of the threshold. In addition, significant changes may only be announced when the respective measurements also exceed the assigned thresholds.

5.2 Scenario

Based on EDT, the effects of this approach are exemplified at the introduced fire detection scenario. As a reminder, Figure 5.3 depicts the respective EDT, which was already presented in Figure 4.2. After having determined all indicators i_S for single sensing capabilities, i.e., the EDT-nodes 3, 6 and 9 in this example, the next step is to merge these for complex events. According to the EDT, the indicators are merged at AND and OR tree-nodes. AND nodes take the average i_S values of child nodes, whereas OR nodes apply the maximum i_S of both child nodes. For the sake of completeness, NOT nodes in EDTs directly adopt the i_S of their child node. In the example scenario, the EDT-node 5 gains the average value of the i_S's determined at the EDT-nodes 6 and 9. Finally, the i_S of the complex event fire, which is attached to the root node, is the maximum of the i_S values from the carbon monoxide readings (EDT-node 2) and the average of the smoke and temperature i_S values (EDT-node 5).

The simulations applied two large-scale data sets (SDC07 and SDC08)[1] from series of real fire tests in a house equipped with one sensor in every room, which have been recorded by the National Institute of Standards and Technology (NIST) [45]. Amongst other things, these sets contain temperature, smoke and carbon monoxide sensor readings[2]. Here, one set of data of a flaming fire and one of a smouldering fire were used. A sliding window size of ten was applied to determine the variance of sensor readings of the last 40 seconds because of an event evaluation rate of four seconds. The i_S-based approach and the usual de-

[1] http://smokealarm.nist.gov/

[2] The simulations results presented in Chapter 4 would had greatly benefit from the availability of such large sets of data of real deployments. Unfortunately, the number of sensors and the density of those is far from being enough to simulate a WSN and to show the abilities of using the EDTs and voting.

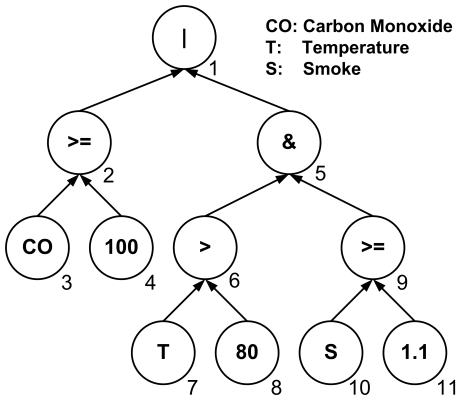


Figure 5.3: EDT for a fire fighting system using carbon monoxide (CO), smoke (S) and temperature (T) detectors. This is just to remind the EDT already introduced in Figure 4.2.

tection by thresholds were simulated at two different sensors, one sensor was directly located above the monitored fire and one sensor was located in a side room. The simulation results of both approaches are presented in Figure 5.4. It shows the local detection results of the two sensors in the flaming fire scenario, see the diagrams 5.4(a) and 5.4(b), and the smouldering fire scenario in the diagrams 5.4(c) and 5.4(d). The threshold-based detection distinguishes between two states, which is 1/TRUE for an exceeded threshold (fire alarm) and 0/FALSE otherwise (no alarm). The i_S-based approach only signals significant changes in sensor readings, but in both scenarios it indicates the upcoming fire event much earlier than the thresholds are exceeded. In particular, the i_S-based approach indicates the flaming fire 88 seconds and the smouldering fire 48 seconds before the threshold-based method triggers the alarm. Thus, the i_S-based detection offers great advantages and gains valuable time for fire fighting systems. The difference between both methods is less in the smouldering fire scenario. Here the detection almost solely depends on the smoke readings, which increase very slowly and hence, are less recognised by the indicator-based approach. According to this, a proper combination of both detection methods is the safest solution, especially for mission- and safety-critical applications. Note, the slow increase is also the reason for requiring more than the double monitoring time before the smouldering fire is detected.

However, the current results strongly indicate a great benefit in event detection quality. On the one hand, the i_S can signal significant changes in sensor readings before these reach a critical point. Thus, necessary processes or further analyses can be triggered in sufficient time. On the other hand, the i_S is even fully functional without predefined thresholds. Thresholds are introduced to assign a higher weight to event relevant readings. Using those thresholds results in

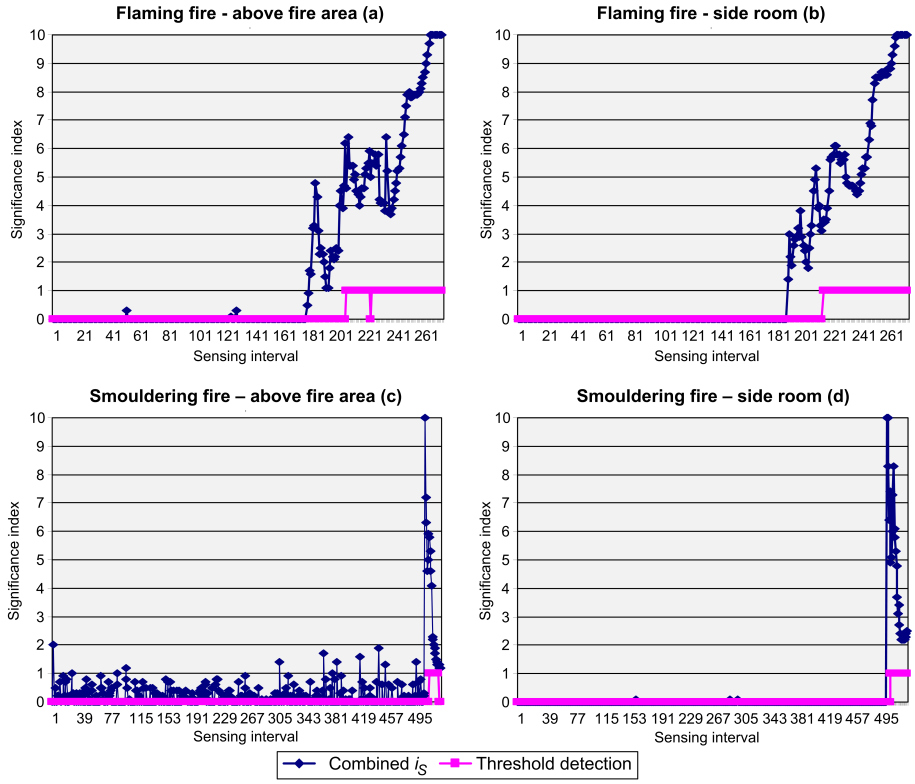


Figure 5.4: Local detection results of the flaming fire scenario at the sensor above the fire area (a) and in the side room (b). For the smouldering fire scenario, (c) displays the readings of the sensor above the fire area and (d) the respective readings in the side room. The i_S-based approach signals significant changes in sensor readings earlier than the thresholds are exceeded. Hence, this approach indicates upcoming fires before the fire alarm is triggered.

a higher and more noticeable amplitude. To give a proof of concept, the four scenarios have been simulated again without predefined thresholds. The results are given in Figure 5.5. In comparison to 5.4(a), diagram 5.5(a) shows the calculation results at the same sensor with equal inputs but without considering thresholds. The same applies to the other diagrams 5.4(b) and 5.5(b), 5.4(c) and 5.5(c) and 5.4(d) and 5.5(d). The fire events are still clearly visible even without predefined thresholds, especially in the smouldering fire scenarios (c) and (d). The number of available test results is yet insufficient to announce the i_S-based approach as a suitable stand-alone event detection method, but the good initial results may be confirmed when evaluating further applications.

Despite all good results, there is a short remark necessary. Running the simulations with the provided test data discovered a small pitfall when exactly using the introduced calculation. If sequential measurements are equal for a long time, as it is not unusual for surveillance scenarios, it may occur that the

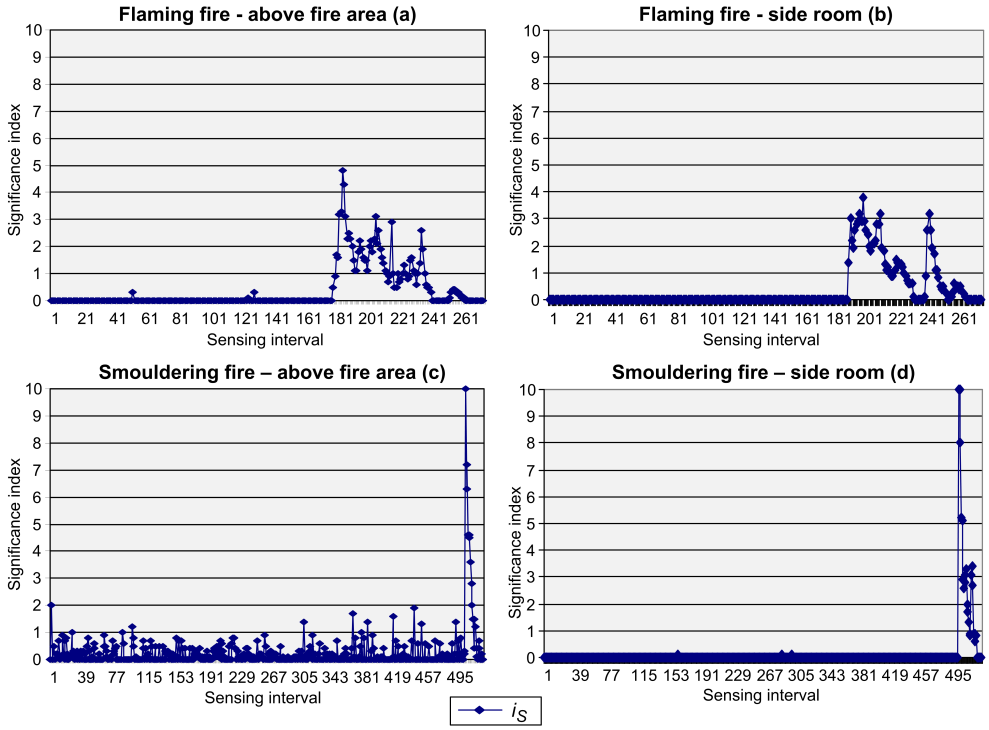


Figure 5.5: Local detection results without predefined thresholds. Please compare the results of (a) to 5.4(a) as well as diagram (b) to 5.4(c). The i_S-based approach also allows to indicate upcoming fire events without predefined thresholds.

resulting variance is extremely small. Depending on the accuracy to be used for calculation, the variance may even become zero in the worst case due to rounding. In the given test data that case occurred during calculation of the i_S for the carbon monoxide readings, which are pretty equal during the initial monitoring phase before the *fire* is triggered. Hence, even small changes in current readings may result in a very high i_S. There exist two possible solutions to cope with that at implementation. The easiest solution is to make sure to only use variables providing enough accuracy for calculation, e.g., by using double precision variables instead of integer. This is obviously not the best solution for WSN since it increases the cost of processing and memory. Instead, a standard deviation used as lower bound for the calculated variance is introduced. Due to the fact that sensor technology always have such standard deviation as potential error of measurement, it is usually given by the manufacturer of the sensing device anyway. Hence it is proposed to include the potential error of measurement as lower bound at implementation. In the given examples the presented problem was properly solved by applying a lower pound of 0.01 ppm.

5.3 Short summary

A behavioural trend analysis of sensor readings can improve the quality of event detection in sensor networks. Therefore a maximum likelihood estimate is adapted to sensor needs by applying the parallel axis theorem and the sliding window principle, to calculate the variance of previous readings. Using the parallel axis theorem gets rid of the necessity to have all previous sensor readings available to calculate the variance. As a result, only two sums determining the sum and the sum of squares of previous readings, have to be stored and updated at each evaluation interval. Since these sums grow fast, a sliding window estimate is applied that limits the number of considered previous readings to the size of the window and hence, it prevents the sums from overflow. The combination of both methods clearly decreases the memory and calculation effort to such extend that it enables even resource-constraint devices like sensor networks to determine the variance over long monitoring phases. Additionally, fixing the size of the sliding window enables the user to properly configure the number of previous readings to be included in calculation.

The variance allows to determine an indicator for the significance of actual measurements i_S that identifies unusual changes in current readings. The i_S is fully independent of the kind of measurement as well as of deployment and application constraints. Hence, the i_S can automatically be applied for every application, even if these are yet unknown. Simulations applying a series of test data from real fire scenarios demonstrated the effectiveness of this approach. A comparison of the simulation results to usual detection methods by thresholds clearly shows that the i_S-based approach can detect significant changes in sensor readings. Moreover, it improves the detection of events and allows for an earlier response in the fire scenario. The i_S indicated the flaming fire 88 seconds and the smouldering fire 48 seconds ahead of the threshold-based method. This offers a great benefit for fire fighting applications. Nevertheless, it is currently proposed to combine both methods for highly reliable detection but the results also indicate a sound basis for the i_S to be solely used for detection.

Chapter 6

Conclusions

This thesis identified missing features of reliable event detection in Wireless Sensor Networks (WSNs). To remedy these shortcomings, it introduces objectives for reliable event-based applications in WSNs in terms of design criteria. These are *Fault tolerance*, *Adaptivity*, *Autonomy*, *Transparency*, *Energy efficiency*, *Convenience*.

Existing work mostly provides fault tolerance and adaptivity but disregards sufficient energy efficiency, autonomy and convenience. It has further been shown that existing fault tolerant solutions lack of means to achieve an acceptable cost efficiency. The envisioned pervasiveness of WSNs faces two major problems. These are high fault probability and configuration complexity. First, pervasive WSNs consisting of large numbers of devices demand to use low cost sensor nodes with limited resources, which feature a high fault probability. Sensing devices attached to the sensor nodes possess certain errors of measurement. Further, WSNs are subject to sudden changes in operational conditions, varying deployments and hazardous environments that again increase the fault probability. Moreover, strict energy constraints on used devices require fault tolerant methods to achieve a high cost efficiency. Second, an ease of use for task definition and configuration of WSNs is the most important issue to make them widely accepted. Means that provide a high abstraction of WSNs are in demand. These must enable also non-professional users, which are usually short on experience of programming languages and sensor networks, to make use of WSNs.

6.1 Contributions

This thesis introduced a novel event concept for definition and configuration of reliable event detection in WSNs. It tackles all design criteria and features cost efficient fault tolerance and a proper usability. So it combines a flexible event def-

inition language with a self-adapting event detection scheme. The Event Specification Language (ESL) provides ease of use for application programming allowing the user to ignore low-level details of the sensor network and to concentrate on a high abstraction level. Namely this is the event itself and its related constraints. To cope with the fault probability in WSNs, cost efficient means for collaborative event detection and evaluation of detection results by voting have been introduced and proven to be functional. In detail, the following contributions are made:

High abstraction for ease of use of event definition. The ESL hides low level details of WSNs to focus on pure event definition, which allows manual or automatic event configuration. The XML-styled language enables to combine sensing features defining the complex phenomena to be sensed. Further, it enhances an event specification by assignment of customised application requirements regarding the spatial and temporal resolution and parameters for voting and collaboration schemes. Especially with regard to voting, the ESL allows to fine tune the voting procedure by determining the voting region, the preferred number of voters and a time limit. Finally, the event description generator transparently processes and adapts event specifications to the target sensor platform.
The ESL addresses the following design criteria: *Fault tolerance*, *Adaptivity*, *Transparency*, *Energy efficiency*, *Convenience*

A novel decentralised mechanism to autonomously set up event detection and in-network processing on sensor nodes, called Event Decision Tree (EDT). Specified event descriptions are deployed on the sensor nodes as EDTs, which are directly generated on the nodes by a tiny GFSM requiring eight states only. An EDT enables the sensor nodes to self-divide event queries according to its own resources into local and remote parts by pruning. Local parts can be evaluated by the node itself whereas values of remote parts must be requested from EDTs at other nodes. Sensor nodes are enabled to maintain several EDTs at the same time. Using EDTs, every node in the network can execute the complete evaluation process without a Single Point of Failure (SPoF).
EDTs address the following design criteria: *Adaptivity*, *Autonomy*, *Transparency*, *Convenience*

Cost efficient means to maintain EDTs in case of missing or failing sensing devices. The EDTs are enabled to continue event detection with a high accuracy even in case of missing resources or failed sensing devices. For those cases, EDTs provide efficient collaborative event detection between neighboring nodes using a lease-based publish/subscribe approach. Appropriate on-node processing of sensed data allows to efficiently share event information by a few bytes only.

The simulations clearly announced that the cost-efficiency of the lease-based approach is very high in contrast to the ACK-based variant, which provides the best detection accuracy. By choosing a proper leasing time, the lease-based approach closely meets the detection results of ACK-based collaboration but reduces the number of collaboration messages by a factor of 20 or higher. The leasing time is set by the user in the event specification. As learnt from simulations, a proper leasing time meets both of the following restrictions:

1. The maximum leasing time is less or equal to the mean time of exposure to the phenomenon to be sensed.

2. The maximum leasing time is less than the mean time to failure.

The lease-based publish/subscribe approach addresses the following design criteria: *Fault tolerance*, *Adaptivity*, *Autonomy*, *Transparency*, *Energy efficiency*, *Convenience*

Reactive Majority Voting (RMV) to cope with uncertainty in sensor readings. Voting is a proper means to enhance the reliability of event detection and to cope with deviating sensor readings but introduces an overhead in time and communication. To reduce this overhead, RMV locally triggers a voting only to evaluate detected events. By that, RMV rejects *False positives* and isolates the nodes that most probably correctly detected an event. This is sufficient for monitoring scenarios such as fire detection. Voting only in case of events reduces the number of voting messages and hence, significantly improves the cost efficiency. To avoid a fixed node collecting all votes as a SPoF, RMV applies unfixed local voting regions around the nodes to independently apply voting. The accuracy of voting highly depends on the size of that voting region. The size of the voting region can be customised in the event specification. To ease the configuration of voting for non-professional users, they should apply the following principles:

1. The minimum size of the voting region is the mean distance between neighbouring sensor nodes, which is determined by the density of the sensor network, and transmission technology.

2. The maximum size of the voting region is the estimated size of the phenomenon.

RMV addresses the following design criteria: *Fault tolerance*, *Autonomy*, *Energy efficiency*

Indicator for the significance of sensor readings The significance indicator i_S upgrades the detection facilities of WSNs applying EDTs. It statistically examines behavioural trends in sensor readings to indicate the significance of actual

measurements in relation to the configured event description. The i_S is independent from the kind of sensor readings and is efficient in calculation and memory effort. The significance indicator i_S can automatically be attached to each event to support users or overlaying systems in decision-making. In the example scenario based on data of real test cases, i_S indicates a flaming fire 88 seconds and a smouldering fire 48 seconds before the threshold-based method triggers the alarm. This concept addresses the following design criteria: *Fault tolerance*, *Energy efficiency*, *Convenience*

To summarise, this thesis presented and evaluated means to enhance the fault tolerance and reliability of event-based application in WSNs. As a final result, criteria for proper event definition were deduced from the simulation results. These criteria ease the configuration of a proper voting region and leasing time by definition of lower and upper bounds.

6.2 Future work

Future work primarily includes the application and test of the ESL and respective EDTs in real world scenarios like Ambient Assisted Living (AAL) or patient monitoring. This also stresses the designed support for mobility in this approach. The ESL allows to easily customise and configure a Body Area Network (BAN) to the needs of a patient, e.g., by configuring thresholds for blood pressure or body temperature. In AAL applications the user may define personal interests as an event specification, which can be used to configure the local ambient sensor network around the user.

Some extensions of the ESL are possible and of interest for further research. The ESL may also allow supporting other voting algorithms to cover a wider range of potential failures, e.g., malicious nodes or Byzantine faults. In view of execution constraints, means should be integrated that allow configuring resource-oriented execution intervals. This could be scaling of the event evaluation interval due to drained energy resources. In addition, the XML-style of the ESL should allow to provide configuration means for WSNs by the use of web technologies and web services. This may automate the configuration process and enable remote configuration via the Internet.

The concept of EDT has to be enhanced by transportable versions of EDTs to enable self-configuration of new nodes during runtime. This may significantly improve the maintenance of a sensor network by allowing newly deployed nodes to populate with the EDTs from their neighbouring nodes. This should allow to easily rebuild the sensor network in areas where nodes have crashed or the node density has to be increased. Furthermore, the fault tolerant methods of voting and collaboration could also be extended. Sensor nodes could dynamically resize

the applied voting region with respect to the local node density, the specified voting region and the determined preferred number of voters. For random distributions, the nodes in areas with low density may use the maximum configured voting region. In contrast to that, the nodes in areas with a high density may apply smaller voting regions, which still provide a sufficient number of available voters. The simulations indicated that the lease-based publish/subscribe approach may possibly hold EDT-node values even if the respective sensing device of the publisher has failed during the leasing time. In the simulation scenarios, this caused detection of *False positives*. Signalling failed devices to subscribing sensor nodes may possibly further enhance the detection accuracy.

For i_S future work has to evaluate whether a complete waiver of thresholds still suitably supports reliable event detection based on this indicator. In addition, the usage of redundant data sources should be considered in future work to develop a distributed variant of i_S. Analogue to the principles and effects of voting, this may protect against faults on single devices and further enhance the reliability of detection when using i_S. The i_S can of course be applied automatically to all kind of readings by determining the size of the sliding window only. An adaptation process for automatic configuration of a proper sliding window size would provide ease of use for application programmers. The introduced surveillance scenarios mostly imply a fixed deployment of sensor nodes. Future projects should investigate also mobile scenarios, e.g., the usage of BANs for customised patient monitoring or portable surveillance systems. The goal is to use the i_S for autonomous detection under varying application and deployment constraints. This stresses the determination and correct evaluation of behavioural trends in changing sensor readings. Last but not least, the i_S shall also be included into the ESL. Automatic application of the i_S for certain event descriptions can be easily achieved by adding another attribute to the event element. This attribute may further determine the size of the applied sliding window. However, this requires to exactly specify the effects of determined i_S values on the evaluation process and the final evaluation results. Automatic application should not be integrated before the potential influences of i_S are further investigated as mentioned.

Bibliography

[1] M. Aboelaze and F. Aloul. Current and future trends in sensor networks: A survey. In *Proc. Second IFIP International Conference on Wireless and Optical Communications Networks WOCN 2005*, pages 551–555, 2005. [cited at p. 9]

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38:393–422, 2002. [cited at p. 9]

[3] P. Bahl and V.N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *Proc. IEEE Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2000*, volume 2, pages 775–784 vol.2, 2000. [cited at p. 34]

[4] M. Bahrepour, N. Meratnia, and P.J.M. Havinga. Sensor fusion-based event detection in wireless sensor networks. In *Proceedings of the Third International Workshop on Information Fusion and Dissemination in Wireless Sensor Networks (SensorFusion09)*, Toronto, Canada, July 13-16 2009. [cited at p. 39]

[5] Danilo Beuche. *Composition and Construction of Embedded Software Families*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2004. [cited at p. 43]

[6] Chatschik Bisdikian, Joel Branch, Kin K. Leung, and Robert I. Young. A letter soup for the quality of information in sensor networks. In *IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, pages 1–6, Los Alamitos, CA, USA, 2009. IEEE Computer Society. [cited at p. 23]

[7] J. Blumenthal, F. Reichenbach, and D. Timmermann. Minimal transmission power vs. signal strength as distance estimation for localization in wireless sensor networks. In *3rd IEEE International Workshop on Wireless Ad-hoc and Sensor Networks , IWWAN 2006*, New York, USA, June 2006. [cited at p. 34]

[8] Jan Blumenthal, Dirk Timmermann, Carsten Buschmann, Stefan Fischer, Jochen Koberstein, and Norbert Luttenberger. Minimal transmission power as distance estimation for precise localization in sensor networks. In *IWCMC '06: Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pages 1331–1336, New York, NY, USA, 2006. ACM. [cited at p. 34]

[9] Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, and Michael Rohs. Living in a world of smart everyday objects – social, economic, and ethical

implications. *Journal of Human and Ecological Risk Assessment*, 10:763–786, 2004. [cited at p. 9]

[10] Nirupama Bulusu, John Heidemann, and Deborah Estrin. Gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications*, 7(5):28 – 34, Oct 2000. [cited at p. 34]

[11] Phil Buonadonna, David Gay, Joseph M. Hellerstein, Wei Hong, and Samuel Madden. Task: Sensor network in a box. In *In Proceedings of European Workshop on Sensor Networks*, pages 133–144, Istanbul, Turkey, 2005. [cited at p. 15, 20, 72]

[12] B. Carbunar, A. Grama, J. Vitek, and O. Carbunar. Coverage preserving redundancy elimination in sensor networks. In A. Grama, editor, *Proc. First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks IEEE SECON 2004*, pages 377–386, 2004. [cited at p. 16]

[13] Mihaela Cardei, Shuhui Yang, and Jie Wu. Algorithms for fault-tolerant topology in heterogeneous wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(3), March 2008. [cited at p. 14]

[14] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, November 2002. [cited at p. 38]

[15] Alexandru Coman, Mario A. Nascimento, and Jörg Sander. Exploiting redundancy in sensor networks for energy efficient processing of spatiotemporal region queries. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 187–194, New York, NY, USA, 2005. ACM. [cited at p. 16]

[16] M.D. Dikaiakos, A. Florides, T. Nadeem, and L. Iftode. Location-aware services over vehicular ad-hoc networks using car-to-car communication. *IEEE Journal on Selected Areas in Communications*, 25(8):1590–1602, October 2007. [cited at p. 9]

[17] Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In *Proc. of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, Boulder, Colorado, USA, November 2006. [cited at p. 27]

[18] Federal Standard 1037C. *Telecommunications: Glossary of Telecommunication Terms*. General Services Administration (GSA), 1996. [cited at p. 9]

[19] O. Garcia Morchon, T. Falck, T. Heer, and K. Wehrle. Security for pervasive medical sensor networks. In *6th Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous 2009)*, Toronto, Canada, July 2009. [cited at p. 9]

[20] E. Gelenbe and L. Hey. Quality of information: An empirical approach. In *5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2008)*, pages 730–735, October 2008. [cited at p. 23]

[21] H.-W. Gellersen, A. Schmidt, and M. Beigl. Adding some smartness to devices and everyday things. In *Proc. Third IEEE Workshop on. Mobile Computing Systems and Applications*, pages 3–10, 2000. [cited at p. 9]

[22] Duncan Gillies, David J. Thornley, and Chatschik Bisdikian. Probabilistic approaches to estimating the quality of information in military sensor networks. *The Computer Journal*, 2009. [cited at p. 9, 23, 110]

[23] I. Haroun, I. Lambadaris, and R. Hafez. Building wireless sensor networks. *Microwave & RF Magazine*, September 2005. [cited at p. 33]

[24] Hossam Hassanein and Jing Luo. Reliable energy aware routing in wireless sensor networks. In *DSSNS '06: Proceedings of the Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems*, pages 54–64, Washington, DC, USA, 2006. IEEE Computer Society. [cited at p. 61]

[25] Douglas Herbert, Vinaitheerthan Sundaram, Yung-Hsiang Lu, Saurabh Bagchi, and Zhiyuan Li. Adaptive correctness monitoring for wireless sensor networks using hierarchical distributed run-time invariant checking. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 2(3):8, 2007. [cited at p. 39]

[26] Inseok Hwang, Qi Han, and Archan Misra. Mastaq: A middleware architecture for sensor applications with statistical quality constraints. In *PERCOMW '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 390–395, Washington, DC, USA, 2005. IEEE Computer Society. [cited at p. 24]

[27] Pankaj Jalote. *Fault tolerance in distributed systems.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994. [cited at p. 38]

[28] Richard Jones and Rafael Lins. *Garbage collection: algorithms for automatic dynamic memory management.* John Wiley & Sons, Inc., New York, NY, USA, 1996. [cited at p. 58]

[29] J. Kahn, R. Katz, and K. Pister. Emerging challenges: Mobile networking for smart dust. *Journal of Communication and Networks*, 2(3):pp. 188–196, September 2000. [cited at p. 9]

[30] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for "smart dust". In *International Conference on Mobile Computing and Networking (MOBICOM)*, pages 271–278, 1999. [cited at p. 9]

[31] Hideki Kamiya, Hiroshi Mineno, Norihiro Ishikawa, Tomoyuki Osano, and Tadanori Mizuno. Composite event detection in heterogeneous sensor networks. *IEEE/IPSJ International Symposium on Applications and the Internet*, 0:413–416, 2008. [cited at p. 19]

[32] T. Kavitha and D. Sridharan. Security vulnerabilities in wireless sensor networks: A survey. *Journal of Information Assurance and Security*, 5 (1), 2010. [cited at p. 67]

[33] L.A. Klein. A boolean algebra approach to multiple sensor voting fusion. *IEEE Transactions on Aerospace and Electronic Systems*, 29(2):317–327, 1993. [cited at p. 16]

[34] M. Krasniewski, Padma Varadharajan, B. Rabeler, S. Bagchi, and Y.C. Hu. Tibfit: Trust index based fault tolerance for arbitrary data faults in sensor networks. In

Proc. International Conference on Dependable Systems and Networks DSN 2005, pages 672–681, 2005. [cited at p. 17, 20, 38, 66, 86]

[35] B. Krishnamachari and S. Iyengar. Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Transactions on Computers*, 53(3):241–250, Mar 2004. [cited at p. 16, 20, 52]

[36] Bhaskar Krishnamachari and S. Sitharama Iyengar. Efficient and fault-tolerant feature extraction in sensor networks. In *2nd Workshop on Information Processing in Sensor Networks, IPSN '03*, Palo Alto, California, April 2003. [cited at p. 16, 20]

[37] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. In W. Weber, J. Rabaey, and E. Aarts, editors, *Ambient Intelligence*. Springer-Verlag, 2005. Springer-Verlag. [cited at p. 15]

[38] Xuanwen Luo, Ming Dong, and Yinlun Huang. On distributed fault-tolerant detection in wireless sensor networks. *IEEE Transactions on Computers*, 55(1):58–70, 2006. [cited at p. 16]

[39] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122 – 173, 2005. [cited at p. 15, 20]

[40] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proc. of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, New York, NY, USA, 2002. ACM. [cited at p. 9]

[41] Jean-Philippe Martin and Lorenzo Alvisi. *Fast Byzantine Consensus*, volume 3. IEEE Computer Society Press, Los Alamitos, CA, USA, 2006. [cited at p. 38]

[42] F. Martincic and L. Schwiebert. Distributed event detection in sensor networks. In *Proc. International Conference on Systems and Networks Communications ICSNC '06*, 2006. [cited at p. 18]

[43] E.F. Nakamura, H.A.B.F. de Oliveira, L.F. Pontello, and A.A.F. Loureiro. On demand role assignment for event-detection in sensor networks. In *Proc. 11th IEEE Symposium on Computers and Communications ISCC '06*, pages 941–947, 26–29 June 2006. [cited at p. 19]

[44] D. Niculescu and B. Nath. Ad hoc positioning system (aps). In *IEEE Global Telecommunications Conference (GLOBECOM)*, volume 5, pages 2926 – 2931, 2001. [cited at p. 34]

[45] NIST Technical Note 1455-1. Performance of home smoke alarms analysis of the response of several available technologies in residential fire settings. Technical report, National Institute of Standards and Technology (NIST), February 2008. [cited at p. 111]

[46] Steffen Ortmann. Definition and configuration of reliable event detection in heterogeneous wsn. In *Third Annual Google Ph.D. Forum on Pervasive Computing and*

Communications in conjunction with the 8th IEEE International Conference on Pervasive Computing and Communications (PerCom) 2010, Mannheim, Germany, March 29 - April 2 2010. [cited at p. 25]

[47] Steffen Ortmann and Peter Langendörfer. Enhancing reliability of sensor networks by fine tuning their event observation behavior. In *Proc. of the 2nd IEEE WoWMoM - Workshop on Adaptive and DependAble Mobile Ubiquitous Systems (ADAMUS'08)*, Newport Beach, CA, USA, June 23 2008. [cited at p. 25]

[48] Steffen Ortmann, Michael Maaser, and Peter Langendörfer. Self-adapting event configuration in ubiquitous wireless sensor networks. *International Journal on Adaptive Resilient and Autonomic Systems (IJARAS), Special Issue on the ADAMUS workshop.* (to appear). [cited at p. 47]

[49] Steffen Ortmann, Michael Maaser, and Peter Langendörfer. Adaptive pruning of event decision trees for energy-efficient collaboration in event-driven wsn. In *Proceedings of the Third International Workshop on Information Fusion and Dissemination in Wireless Sensor Networks (SensorFusion09)*, Toronto, Canada, July 13-16 2009. [cited at p. 47]

[50] A.S.K. Pathan, Hyung-Woo Lee, and Choong Seon Hong. Security in wireless sensor networks: issues and challenges. In *Proc. 8th International Conference Advanced Communication Technology ICACT 2006*, volume 2, pages 6 pp.–1048, 2006. [cited at p. 67]

[51] S. Peter, P. Langendörfer, and Piotrowski K. Public key cryptography empowered smart dust is affordable. *Special issue on Energy-Efficient Algorithm and Protocol Design in Sensor Networks, International Journal of Sensor Networks (IJSNET)*, 4 (1/2), 2008. [cited at p. 67]

[52] S. Peter, O. Stecklina, and P. Langendörfer. An engineering approach for secure and safe wireless sensor and actuator networks for industrial automation systems. In *14th IEEE International Conference on Emerging Techonologies and Factory Automation (ETFA 2009)*, September 2009. [cited at p. 67]

[53] Hai N. Pham, Dimosthenis Pediaditakis, and Athanassios Boulis. From simulation to real deployments in wsn and back. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007.*, pages 1 – 6, June 2007. [cited at p. 68]

[54] A V U Phani Kumar, Adi Mallikarjuna Reddy V, and D. Janakiram. Distributed collaboration for event detection in wireless sensor networks. In *MPAC '05: Proc. of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, pages 1–8, New York, NY, USA, 2005. ACM. [cited at p. 19]

[55] Krzysztof Piotrowski, Peter Langendörfer, and Steffen Peter. How public key cryptography influences wireless sensor node lifetime. In *SASN '06: Proc. of the fourth ACM workshop on Security of ad hoc and sensor networks*, pages 169–176, New York, NY, USA, 2006. ACM. [cited at p. 32]

[56] H. Rangarajan and J.J. Garcia-Luna-Aceves. Reliable data delivery in event-driven wireless sensor networks. In *Proc. Ninth International Symposium on Computers and Communications ISCC 2004*, volume 1, pages 232–237, 28 June–1 July 2004. [cited at p. 61]

[57] Philip Robinson and Michael Beigl. Trust context spaces: An infrastructure for pervasive security in context-aware environments. In *First International Conference of Security in Pervasive Computing*, pages 157–172, 2003. [cited at p. 9]

[58] K. Romer and F. Mattern. The design space of wireless sensor networks. *IEEE Wireless Communications*, 11(6):54–61, 2004. [cited at p. 41]

[59] K. Romer and F. Mattern. Event-based systems for detecting real-world states with sensor networks: a critical analysis. In *Proc. the 2004 Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 389–395, 2004. [cited at p. 10]

[60] Yogesh Sankarasubramaniam, zgr B. Akan, and Ian F. Akyildiz. Esrt: event-to-sink reliable transport in wireless sensor networks. In *The ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, pages 177–188. ACM, 2003. [cited at p. 61]

[61] S. Schwiderski-Grosche. Context-dependent event detection in sensor networks. In *2nd Intl. Conf. on Distributed Event-Based Systems (DEBS'08)*, Rome, Italy, July 2008. [cited at p. 20]

[62] Kuei-Ping Shih, Sheng-Shih Wang, Pao-Hwa Yang, and Chau-Chieh Chang. Collect: Collaborative event detection and tracking in wireless heterogeneous sensor networks. In *Proc. 11th IEEE Symposium on Computers and Communications ISCC '06*, pages 935–940, 2006. [cited at p. 10, 18, 29]

[63] A.P. Speer and I.-R. Chen. Effect of redundancy on mean time to failure of wireless sensor networks. In I.-R. Chen, editor, *Proc. 20th International Conference on Advanced Information Networking and Applications AINA 2006*, volume 2, 2006. [cited at p. 16]

[64] T. Sun, Ling-Jyh Chen, Chih-Chieh Han, and M. Gerla. Reliable sensor networks for planet exploration. In Ling-Jyh Chen, editor, *Proc. IEEE Networking, Sensing and Control*, pages 816–821, Tucson, USA, 2005. [cited at p. 9, 16, 38, 86]

[65] D. J. Thornley, R. I. Young, J. Richardson, and C. Bisdikian. From mission specification to quality of information measures closing the loop in military sensor networks. In *Annual Conference of the International Technology Alliance (ACITA 2008)*, September 2008. [cited at p. 23]

[66] A. Varga. Omnet++. *"Software Tools for Networking", IEEE Network Interactive*, 16(4), 2002. [cited at p. 68]

[67] C.T. Vu, R.A. Beyah, and Yingshu Li. Composite event detection in wireless sensor networks. In *Proc. IEEE International Performance, Computing, and Communications Conference IPCCC 2007*, pages 264–271, 2007. [cited at p. 14, 19]

[68] H. Wada, P. Boonma, and J. Suzuki. A spacetime oriented macroprogramming paradigm for push-pull hybrid sensor networking. In *Proc. 16th International Conference on Computer Communications and Networks ICCCN 2007*, pages 868–875, 2007. [cited at p. 15, 20]

[69] Jürgen Walter. Untersuchung des einflusses unidirektionaler netzwerkverbindungen auf manet- und wsn-routingprotokolle. 2010. [cited at p. 69]

[70] Karsten Walther and Jorg Nolte. A flexible scheduling framework for deeply embedded systems. In *Proc. 21st International Conference on Advanced Information Networking and Applications Workshops AINAW '07*, volume 1, pages 784–791, 2007. [cited at p. 28, 48]

[71] Tsang-Yi Wang, Y.S. Han, P.K. Varshney, and Po-Ning Chen. Distributed fault-tolerant classification in wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4):724–734, 2005. [cited at p. 14, 18]

[72] Mark Weiser. The computer for the twenty-first century. *Scientific American*, 265(3):66 – 75, September 1991. [cited at p. 9]

[73] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Proceeedings of the Second European Workshop on Wireless Sensor Networks*, pages 108–120, 31 Jan.–2 Feb. 2005. [cited at p. 9]

[74] D. Westhoff, J. Girao, and A. Sarma. Security solutions for wireless sensor networks. *NEC Journal of Advanced Technology*, 59(2), June 2006. [cited at p. 67]

[75] A. Willig and H. Karl. Data transport reliability in wireless sensor networks - a survey of issues and solutions. *Praxis der Informationsverarbeitung und Kommunikation*, 28:86–92, April 2005. [cited at p. 23, 61]

[76] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3):9–18, 2002. [cited at p. 15, 20]

[77] Bin Yu and K. Sycara. Learning the quality of sensor data in distributed decision fusion. In *9th International Conference on Information Fusion*, pages 1–8, July 2006. [cited at p. 23, 24]

[78] S. Zahedi, E. Ngai, E. Gelenbe, D. Mylaraswamy, and M.B. Srivastava. Information quality aware sensor network services. In *42nd Asilomar Conference on Signals, Systems and Computers*, pages 1155–1159, Oct. 2008. [cited at p. 23]

Appendices

Appendix A

Event Specification Language (ESL)

A.1 XML schema of the ESL

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="Event" attributeFormDefault="unqualified"
    elementFormDefault="qualified" targetNamespace="Event" xmlns:xs="http:
    //www.w3.org/2001/XMLSchema">

  <xs:element name="EVENT">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="SENSORDATA" maxOccurs="1" minOccurs="1" />
        <xs:element ref="CONSEQUENCE" maxOccurs="1" minOccurs="1" />
        <xs:element ref="EXECUTION" maxOccurs="1" minOccurs="1" />
        <xs:element ref="DIMENSION" minOccurs="0" maxOccurs="1" />
        <xs:element ref="VOTING" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required" />
      <xs:attribute name="version" type="xs:int" use="required" />
      <xs:attribute name="priority" type="priority" use="optional" />
      <xs:attribute name="lease" type="xs:int" use="required" />
      <xs:attribute name="reliableMode" type="mode" use="optional" />
    </xs:complexType>
  </xs:element>

  <xs:element name="SENSORDATA" type="unaryBool" />

  <xs:element name="CONSEQUENCE">
    <xs:complexType>
      <xs:sequence minOccurs="1" maxOccurs="unbounded">
        <xs:element ref="TRIGGERHANDLER" minOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
<xs:element name="EXECUTION">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="TIMEINTERVAL" minOccurs="1" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="DIMENSION" type="spatialResolution" />

<xs:element name="VOTING">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="DIMENSION" minOccurs="0" maxOccurs="1" />
      <xs:element ref="NUMBEROFVOTES" minOccurs="0" maxOccurs="1" />
      <xs:element ref="DEADLINE" minOccurs="0" maxOccurs="1" />
      <xs:element ref="NODEVICES" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="NOT" type="unaryBool" />
<xs:element name="AND" type="binaryBool" />
<xs:element name="OR" type="binaryBool" />

<xs:element name="VARIABLE" type="xs:string" />

<xs:element name="CONSTANT">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:double">
        <xs:attribute name="unit" use="optional" type="Unit">
        </xs:attribute>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="SUM" type="binaryDbl" />
<xs:element name="DIFFERENCE" type="binaryDbl" />
<xs:element name="PRODUCT" type="binaryDbl" />
<xs:element name="QUOTIENT" type="binaryDbl" />
<xs:element name="MODULO" type="binaryDbl" />

<xs:element name="NODEVICES" type="eventConstraint" />
<xs:element name="TIMEINTERVAL" type="eventConstraint" />
<xs:element name="NUMBEROFVOTES" type="eventConstraint" />
<xs:element name="DEADLINE" type="eventConstraint" />
<xs:element name="CIRCLE" type="eventConstraint" />
<xs:element name="SQUARE" type="eventConstraint" />
<xs:element name="BALL" type="eventConstraint" />
<xs:element name="CUBE" type="eventConstraint" />
<xs:element name="HOPS" type="eventConstraint" />
<xs:element name="TRIGGERHANDLER" type="xs:string" />

<xs:complexType name="eventConstraint">
  <xs:sequence>
    <xs:element ref="CONSTANT" maxOccurs="2" minOccurs="1" />
  </xs:sequence>
```

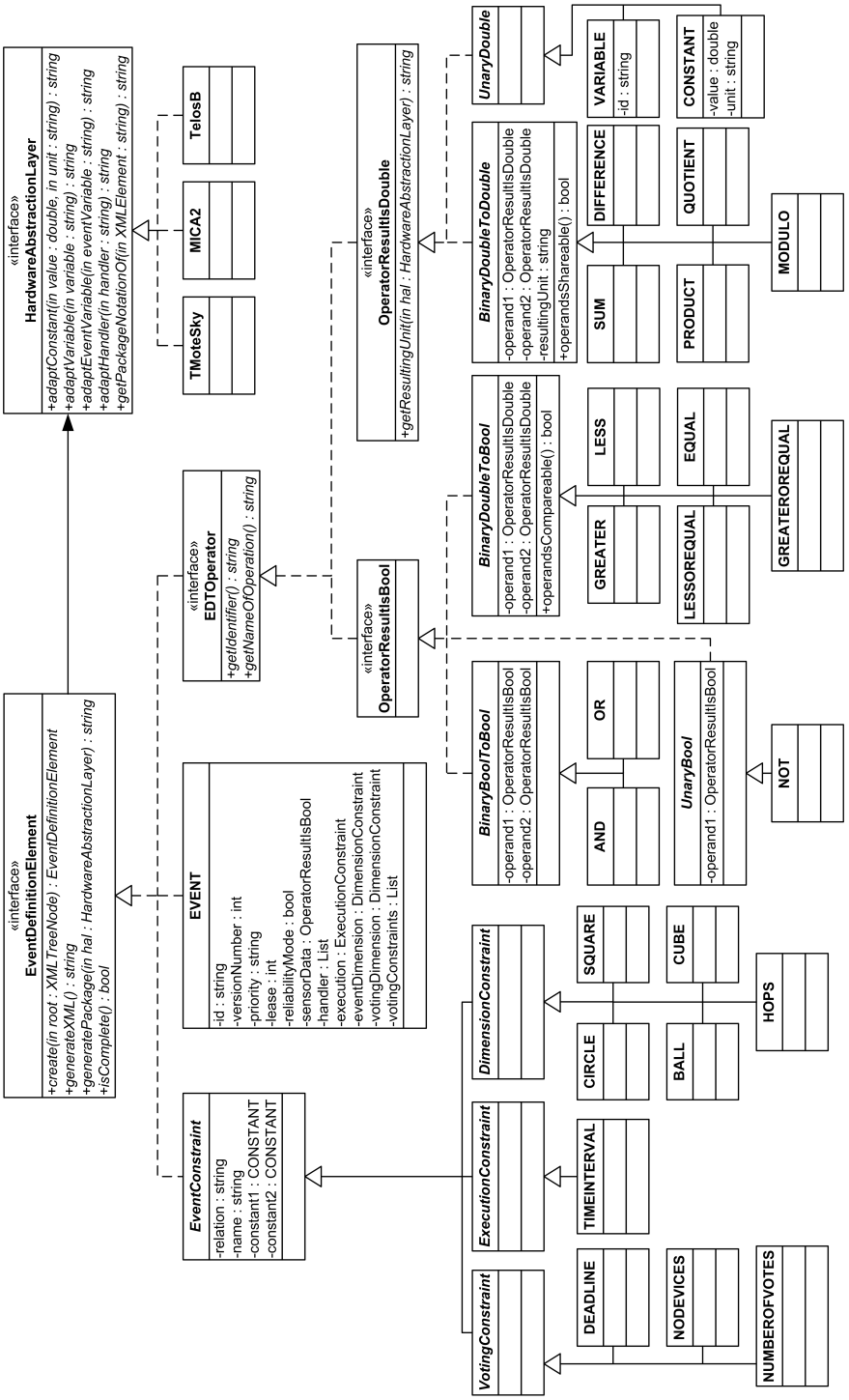
```
  <xs:attribute name="relation" type="Relation" use="required" />
</xs:complexType>

<xs:complexType name="unaryBool">
  <xs:choice minOccurs="1" maxOccurs="1">
    <xs:element ref="NOT" />
    <xs:element ref="AND" />
    <xs:element ref="OR" />
    <xs:element ref="EQUAL" />
    <xs:element ref="GREATER" />
    <xs:element ref="GREATEROREQUAL" />
    <xs:element ref="LESS" />
    <xs:element ref="LESSOREQUAL" />
  </xs:choice>
</xs:complexType>

<xs:complexType name="binaryDbl">
  <xs:choice minOccurs="2" maxOccurs="2">
    <xs:element ref="VARIABLE" />
    <xs:element ref="CONSTANT" />
    <xs:element ref="SUM" />
    <xs:element ref="DIFFERENCE" />
    <xs:element ref="PRODUCT" />
    <xs:element ref="QUOTIENT" />
    <xs:element ref="MODULO" />
  </xs:choice>
</xs:complexType>

<xs:complexType name="binaryBool">
  <xs:choice minOccurs="2" maxOccurs="2">
    <xs:element ref="NOT" />
    <xs:element ref="AND" />
    <xs:element ref="OR" />
    <xs:element ref="EQUAL" />
    <xs:element ref="GREATER" />
    <xs:element ref="LESS" />
    <xs:element ref="GREATEROREQUAL" />
    <xs:element ref="LESSOREQUAL" />
  </xs:choice>
</xs:complexType>

<xs:complexType name="spatialResolution">
  <xs:choice minOccurs="1" maxOccurs="1">
    <xs:element ref="CIRCLE" />
    <xs:element ref="SQUARE" />
    <xs:element ref="BALL" />
    <xs:element ref="CUBE" />
    <xs:element ref="HOPS" />
  </xs:choice>
</xs:complexType>

<xs:element name="EQUAL" type="binaryDbl" />
<xs:element name="GREATER" type="binaryDbl" />
<xs:element name="LESS" type="binaryDbl" />
<xs:element name="LESSOREQUAL" type="binaryDbl" />
<xs:element name="GREATEROREQUAL" type="binaryDbl" />

<xs:simpleType name="Unit">
  <xs:union memberTypes="Distance␣Temperature␣LuminousIntensity␣
```

```
      ElectricCurrent␣Acustics␣Weight␣Time␣RelationalUnit" />
</xs:simpleType>

<xs:simpleType name="Distance">
  <xs:restriction base="xs:string">
    <xs:enumeration value="nanometers" />
    <xs:enumeration value="micrometers" />
    <xs:enumeration value="millimeters" />
    <xs:enumeration value="centimeters" />
    <xs:enumeration value="decimeters" />
    <xs:enumeration value="meters" />
    <xs:enumeration value="kilometers" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="Temperature">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Kelvin" />
    <xs:enumeration value="centigrade" />
    <xs:enumeration value="Fahrenheit" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="LuminousIntensity">
  <xs:restriction base="xs:string">
    <xs:enumeration value="candela" />
    <xs:enumeration value="lux" />
    <xs:enumeration value="lumen" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ElectricCurrent">
  <xs:restriction base="xs:string">
    <xs:enumeration value="ampere" />
    <xs:enumeration value="watts" />
    <xs:enumeration value="volt" />
    <xs:enumeration value="ohm" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="Acustics">
  <xs:restriction base="xs:string">
    <xs:enumeration value="hertz" />
    <xs:enumeration value="pascals" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="Weight">
  <xs:restriction base="xs:string">
    <xs:enumeration value="nanogram" />
    <xs:enumeration value="microgram" />
    <xs:enumeration value="milligram" />
    <xs:enumeration value="gram" />
    <xs:enumeration value="kilogram" />
    <xs:enumeration value="megagram" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="Time">
```

```
    <xs:restriction base="xs:string">
      <xs:enumeration value="nanoseconds" />
      <xs:enumeration value="microseconds" />
      <xs:enumeration value="milliseconds" />
      <xs:enumeration value="seconds" />
      <xs:enumeration value="minutes" />
      <xs:enumeration value="hours" />
      <xs:enumeration value="days" />
      <xs:enumeration value="years" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="RelationalUnit">
    <xs:restriction base="xs:string">
      <xs:enumeration value="percent" />
      <xs:enumeration value="permille" />
      <xs:enumeration value="partsPerMillion" />
      <xs:enumeration value="partsPerBillion" />
      <xs:enumeration value="partsPerTrillion" />
      <xs:enumeration value="partsPerQuadrillion" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="priority">
    <xs:restriction base="xs:string">
      <xs:enumeration value="high" />
      <xs:enumeration value="normal" />
      <xs:enumeration value="low" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="mode">
    <xs:restriction base="xs:string">
      <xs:enumeration value="yes" />
      <xs:enumeration value="no" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Relation">
    <xs:restriction base="xs:string">
      <xs:enumeration value="LessThan" />
      <xs:enumeration value="GreaterThan" />
      <xs:enumeration value="EqualTo" />
      <xs:enumeration value="LessOrEqualTo" />
      <xs:enumeration value="GreaterOrEqualTo" />
      <xs:enumeration value="InBetween" />
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

Listing A.1: XML scheme of the Event Specification Language (ESL)

A.2 Class diagram of the ESL-parser implementation



A.3 State transition table of the EDT-generator.

State	1	2	3	4	5	6	7	8	9
&	2.1	2.1	%	%	%	2.11,1	2.12,1	2.13,1	%
|	2.2	2.2	%	%	%	2.11,2	2.12,2	2.13,2	%
!	2.3	2.3	%	%	%	2.11,3	2.12,3	2.13,3	%
>	4.19	4.19	%	%	%	4.11	4.12	4.13	%
<	5.19	5.19	%	%	%	5.11	5.12	5.13	%
=	2.8	2.8	%	3.6	3.7	2.11,8	2.12,8	2.13,8	%
A..z	%	6.17	6.17	6.4	6.5	6.17	6.12,17	6.13,17	%
0..9	%	7.18	7.18	7.4	7.5	6.17	7.18	9.18	%
+	%	3.9	3.9	3.4,9	3.5,9	3.11,9	3.12,9	3.13,9	%
#	%	3.10	3.10	3.4,10	3.5,10	3.11,10	3.12,10	3.13,10	%
*	%	3.14	3.14	3.4,14	3.5,14	3.11,14	3.12,14	3.13,14	%
/	%	3.15	3.15	3.4,15	3.5,15	3.11,15	3.12,15	3.13,15	%
%	%	3.16	3.16	3.4,16	3.5,16	3.11,16	3.12,16	3.13,16	%
-	%	8.18	8.18	8.4,18	8.5,18	8.11,18	8.12,18	8.13,18	%
.	%	%	%	%	%	%	9.18	%	%
,	%	%	%	%	%	3.11	3.12	3.13	%
:	9.0	%	%	%	%	9.11,0	9.12,0	9.13,0	%
other	%	%	%	%	%	%	%	%	%

Table A.1: State transition table of the Generating Finite State Machine (GFSM) constructing the Event Decision Trees (EDTs). This table present the actions and state transitions performed on sequential single input during parsing of the sensor data element. Each entry specifies a transition in the state machine containing the number of the target state and the actions applied to this transition. At first, the target state is listed followed by a dot and applied actions. Several listed actions are additionally separated by comma. Detailed descriptions about the states and actions applied are separately presented, see Table A.2 for the states and Table A.3 for the actions.

State	Description
1	START
2	Operand or Operation
3	Operand or Arithmetic
4	> or >=
5	< or <=
6	Variable
7	Constant (int)
8	Constant (real)
9	STOP

Table A.2: Description of states applied for the GFSM in Table A.1.

Code	Action
0	STOP and finalise parsed tree
1	Create AND
2	Create OR
3	Create NOT
4	Create GREATER
5	Create LESS
6	Create GREATEROREQUAL
7	Create LESSOREQUAL
8	Create EQUAL
9	Create SUM
10	Create DIFFERENCE
11	Create variable
12	Create constant (int)
13	Create constant (real)
14	Create PRODUCT
15	Create QUOTIENT
16	Create MODULO
17	Collect identifier
18	Collect number
19	Switch state only
%	Abort (error)

Table A.3: Codes and their meanings used for the GFSM in Table A.1.

Appendix B

Diagrams of simulation results

B.1 Simulation results for positive deviating sensor readings

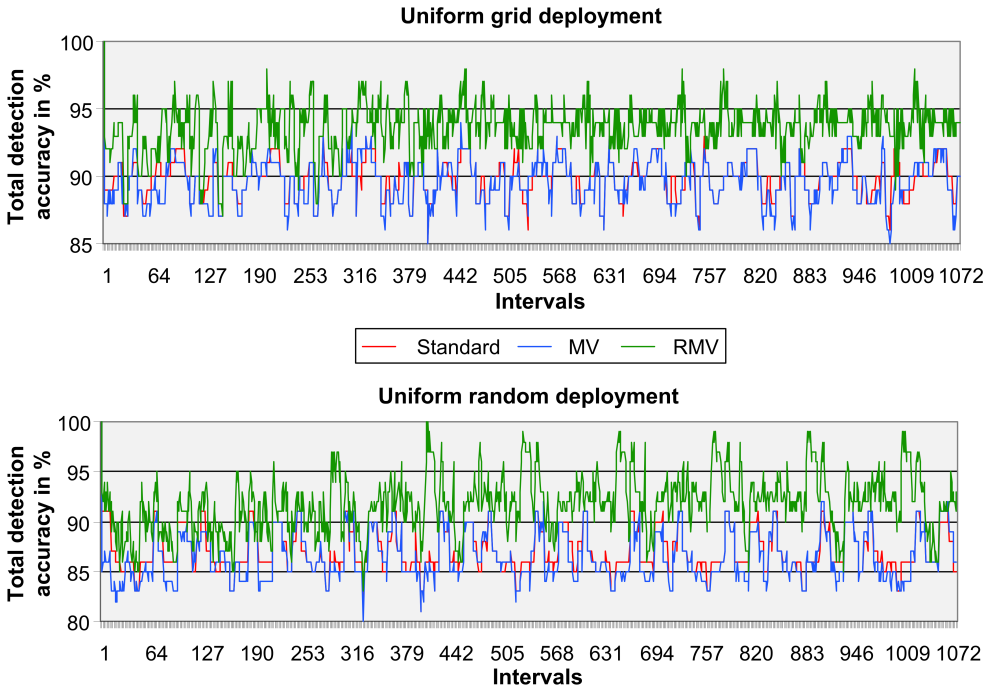


Figure B.1: Comparison of total detection accuracy when applying MV and RMV in case of positive deviating sensor readings. By reducing the number of *False positives*, the RMV approach enhances the accuracy of detection by about 5%. In contrast to that, the MV approach performs nearly equal to the standard detection.

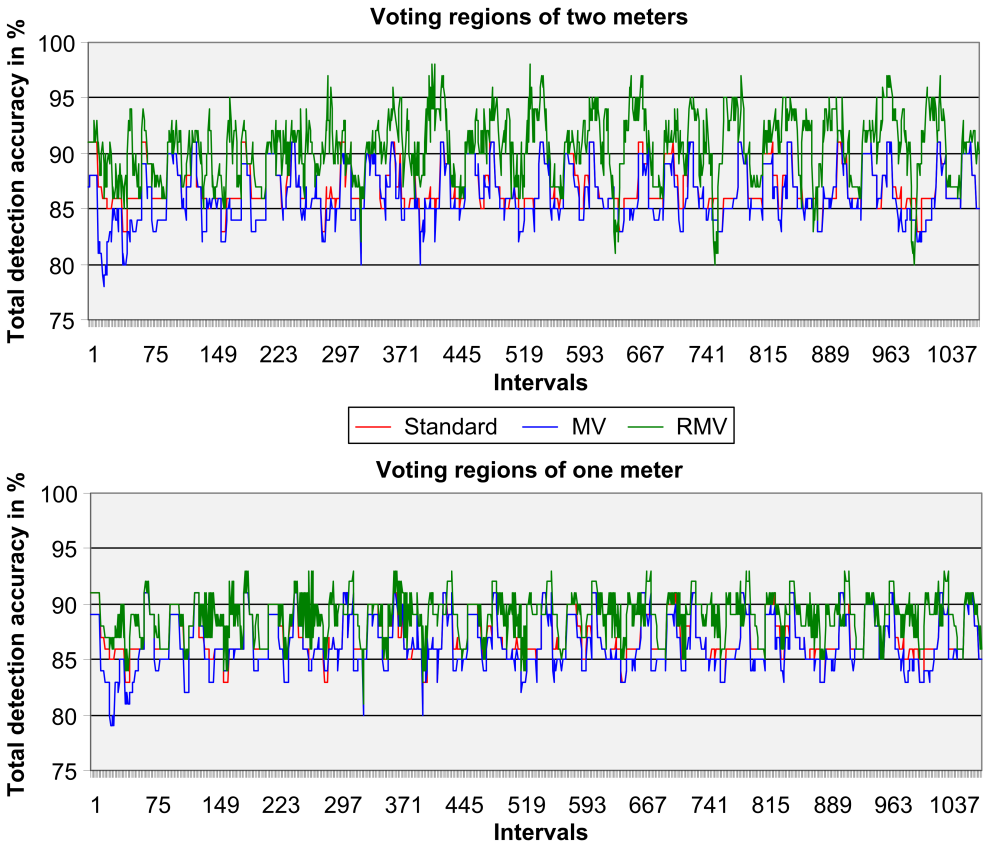


Figure B.2: Comparison of total detection accuracy when applying different voting regions for MV and RMV in case of positive deviating sensor readings. Downsizing the voting region reduces the advantage of RMV to detected less *False positives*. So both voting approaches converge to the standard detection.

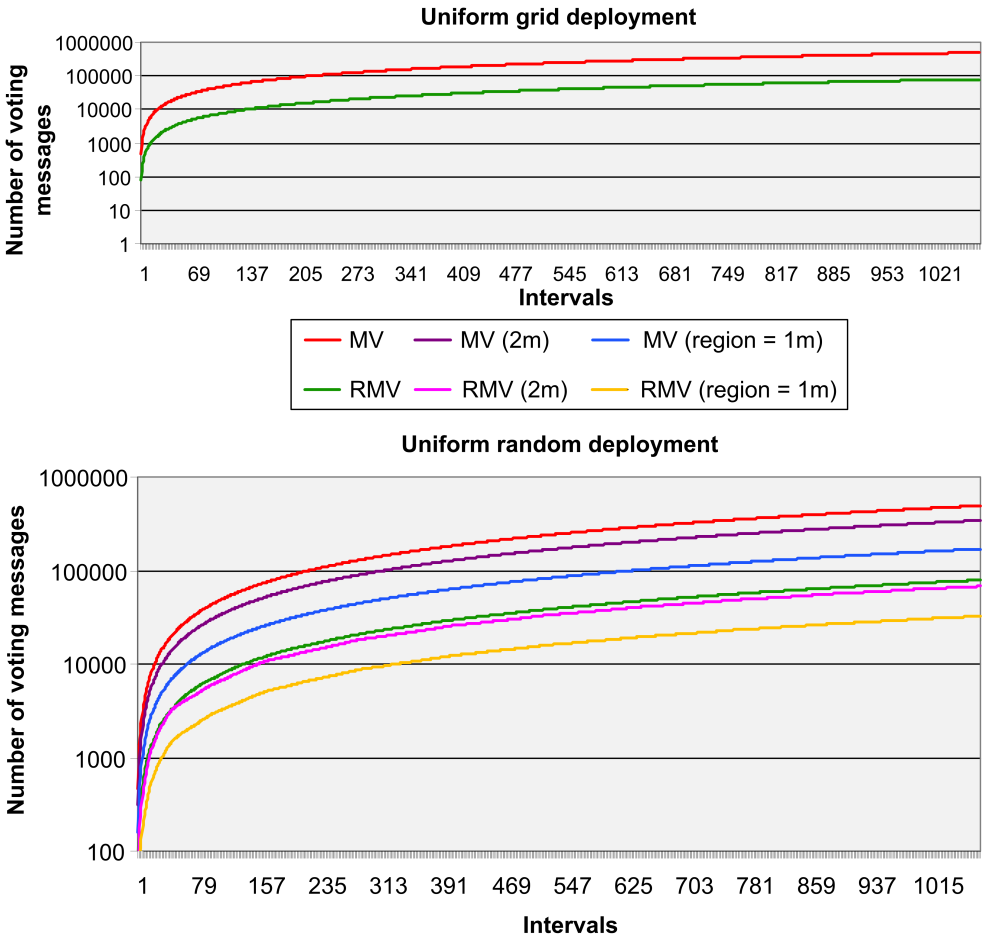


Figure B.3: The overhead of voting is represented by the number of voting messages, which are here given for the entire network. In case of positive deviating sensor readings, the RMV not only increases the detection accuracy but also requires significantly less messages than MV. In comparison to MV, the RMV reduced the average number of messages by a factor of five to six. Please note, these diagrams applied logarithmic scales.

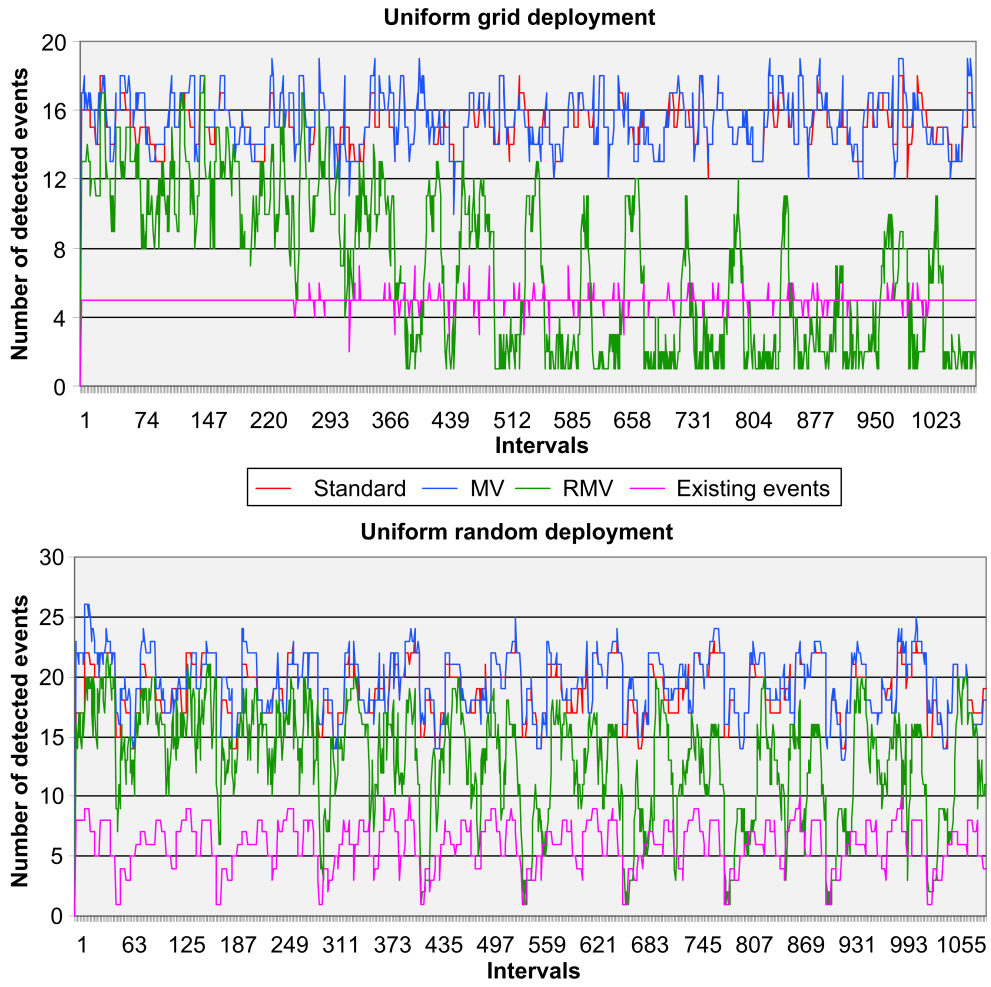


Figure B.4: Number of detected events per interval compared to the existing phenomenon in case of positive deviating sensor readings. The RMV produces only 33% more events in the grid deployment but still doubles the number of detected events in the uniform random deployment. In contrast to that, the standard detection and the MV detect three times more events than actually existed. All approaches detect at least one event per interval and hence, no phenomenon is missed.

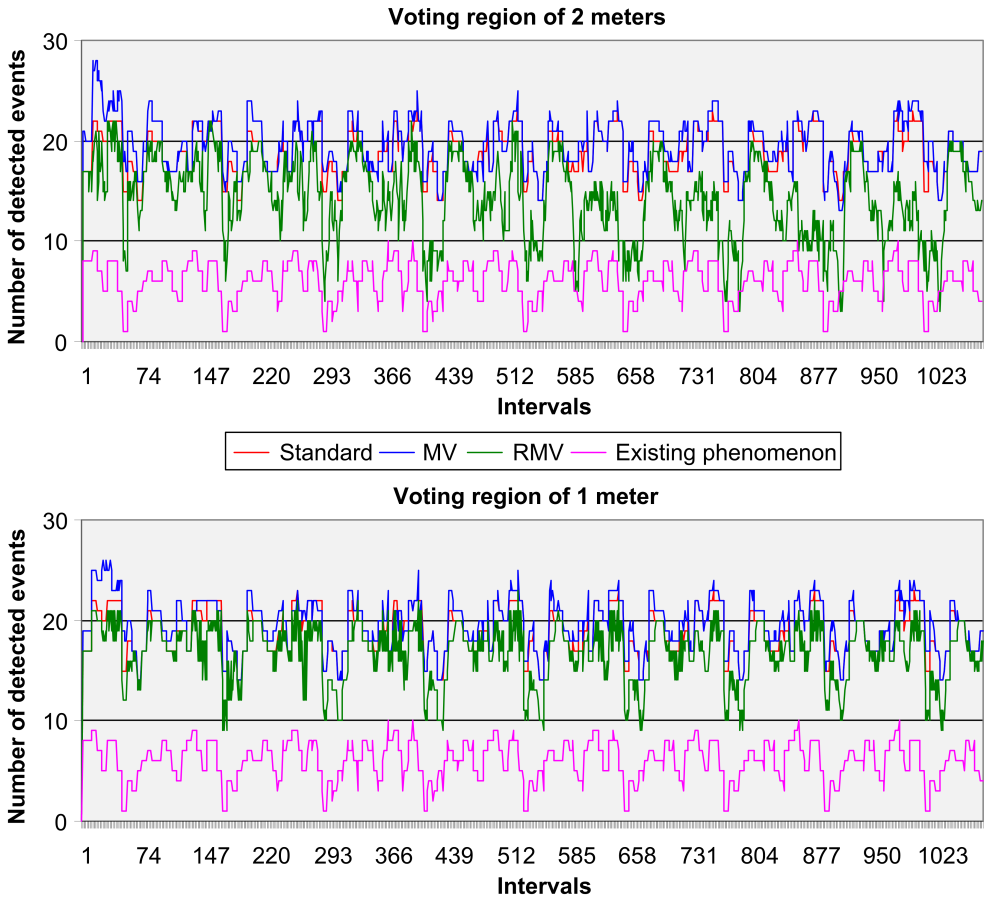


Figure B.5: Number of detected events per interval when applying different voting regions in case of positive deviating sensor readings. With downsizing the voting region the performance of RMV converges to the standard detection, which increases the number of *False positives* due to the less available voters.

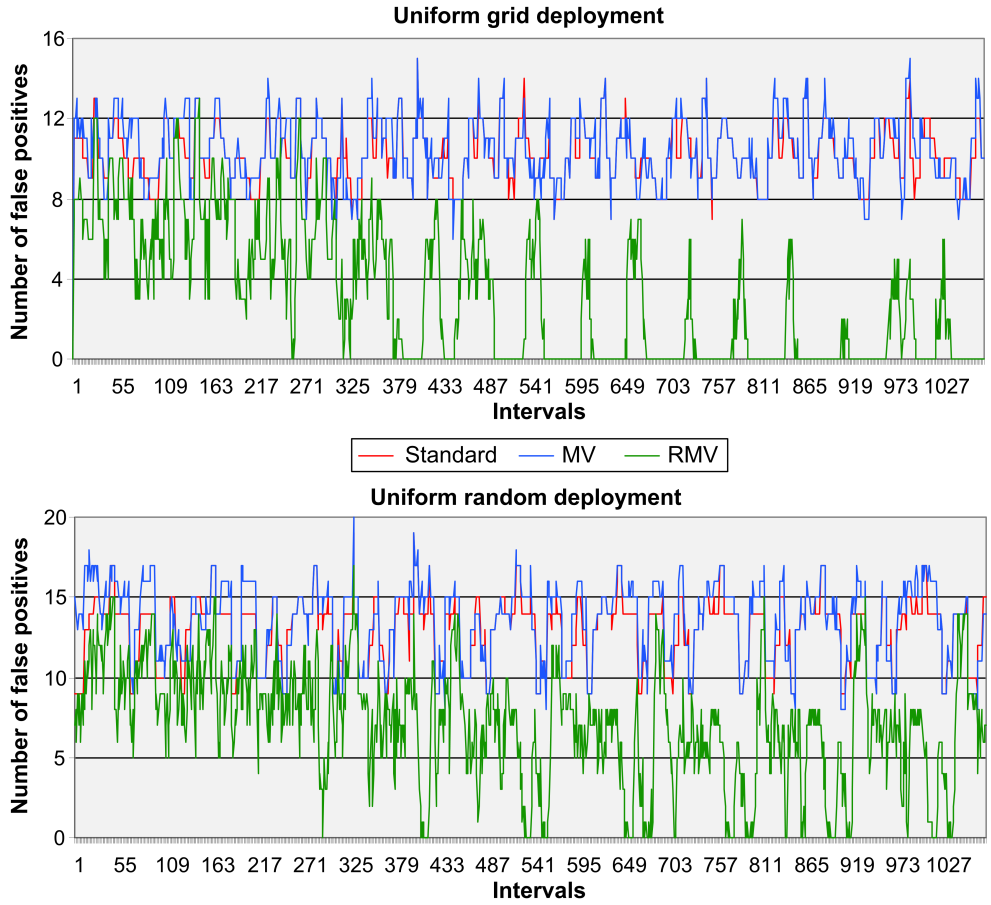


Figure B.6: Comparison of the number of *False positives* per interval between the standard detection and voting. Here, MV and RMV apply a voting region of 2.5 meters. All approaches gather large numbers of *False positives* while RMV at least reduces these to 30% and 55% compared to the standard detection.

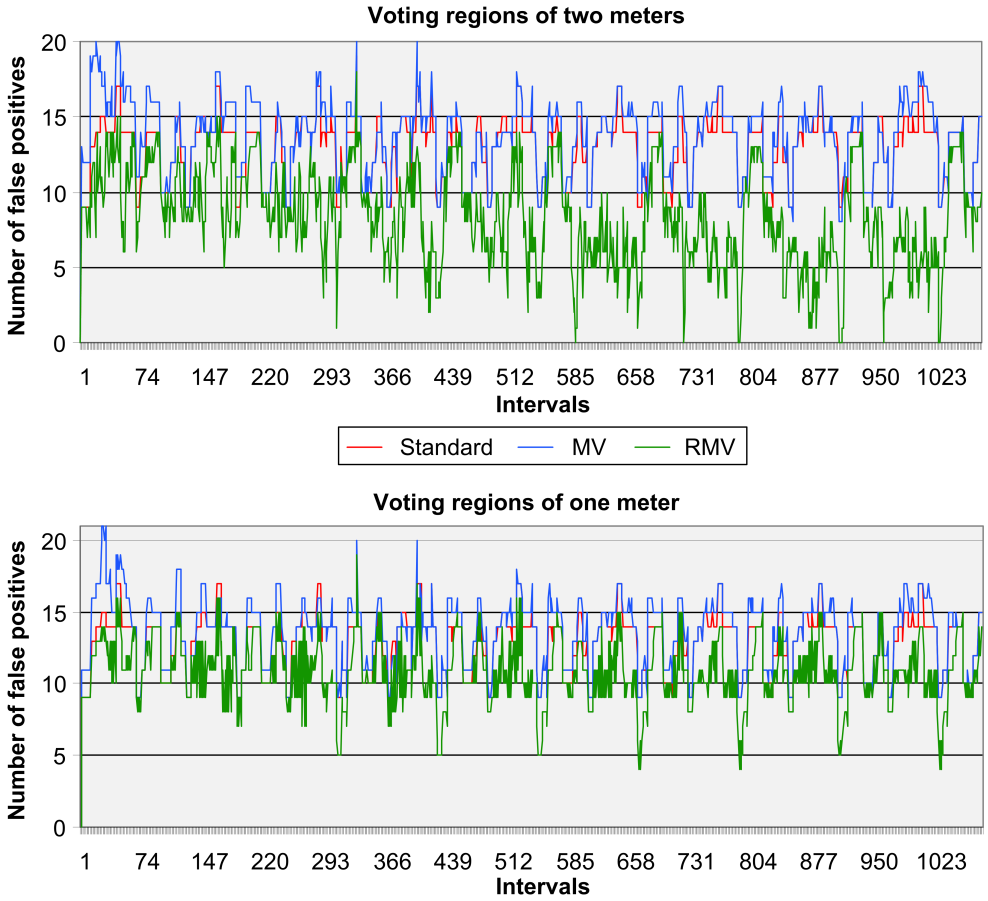


Figure B.7: Comparison of *False positives* per interval between the standard detection and voting. Here, MV and RMV apply voting regions of 2 meters and 1 meter. With downsizing the voting region the number of available voters decreases. For RMV this increases the number of *False positives* due to the fact that less events are overruled by other devices.

B.2 Simulation results for negative deviating sensor readings

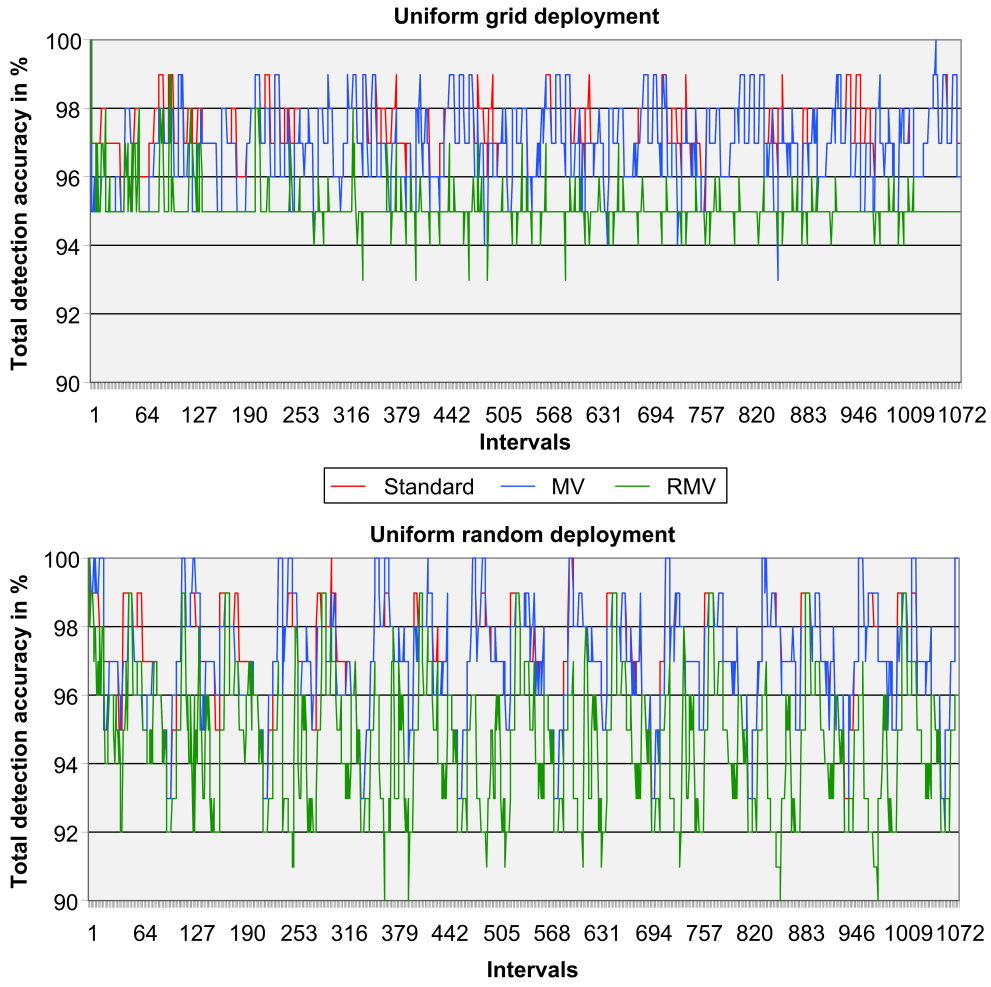


Figure B.8: Comparison of the total detection accuracy when applying MV and RMV in case of negative deviating sensor readings. The standard detection performs best because the voting algorithms tend to overrule and respectively negate the inherently few events.

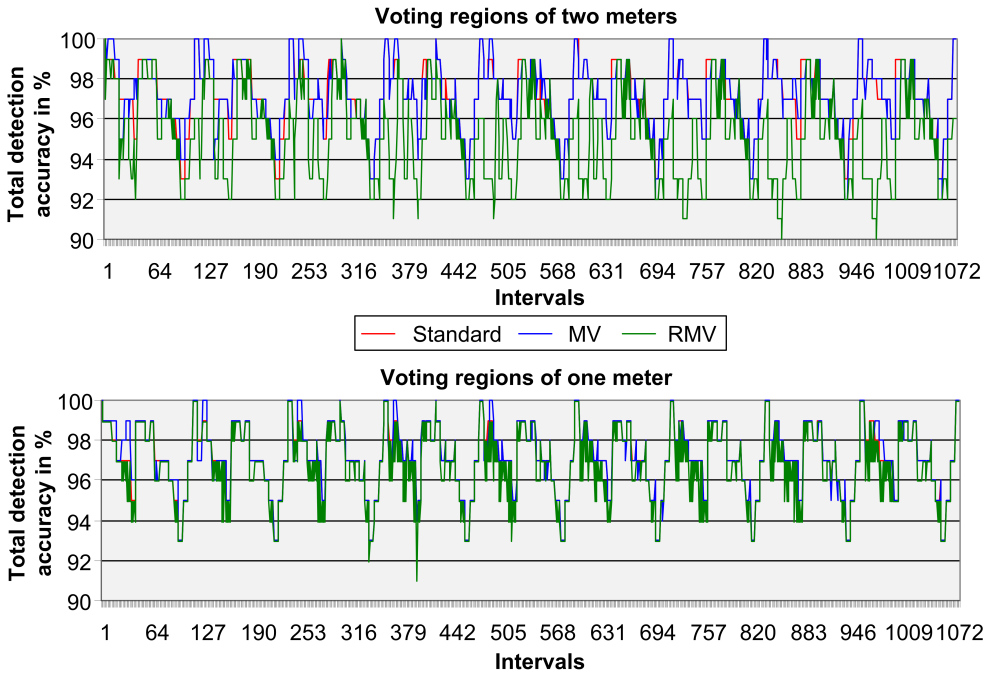


Figure B.9: Comparison of the total detection accuracy when applying smaller voting region for MV and RMV in case of negative deviating sensor readings. Again, with downsizing the voting region the accuracy of both voting approaches increases due to the fact that these converge to the results of the standard detection, but with MV even performing marginally better than the standard detection.

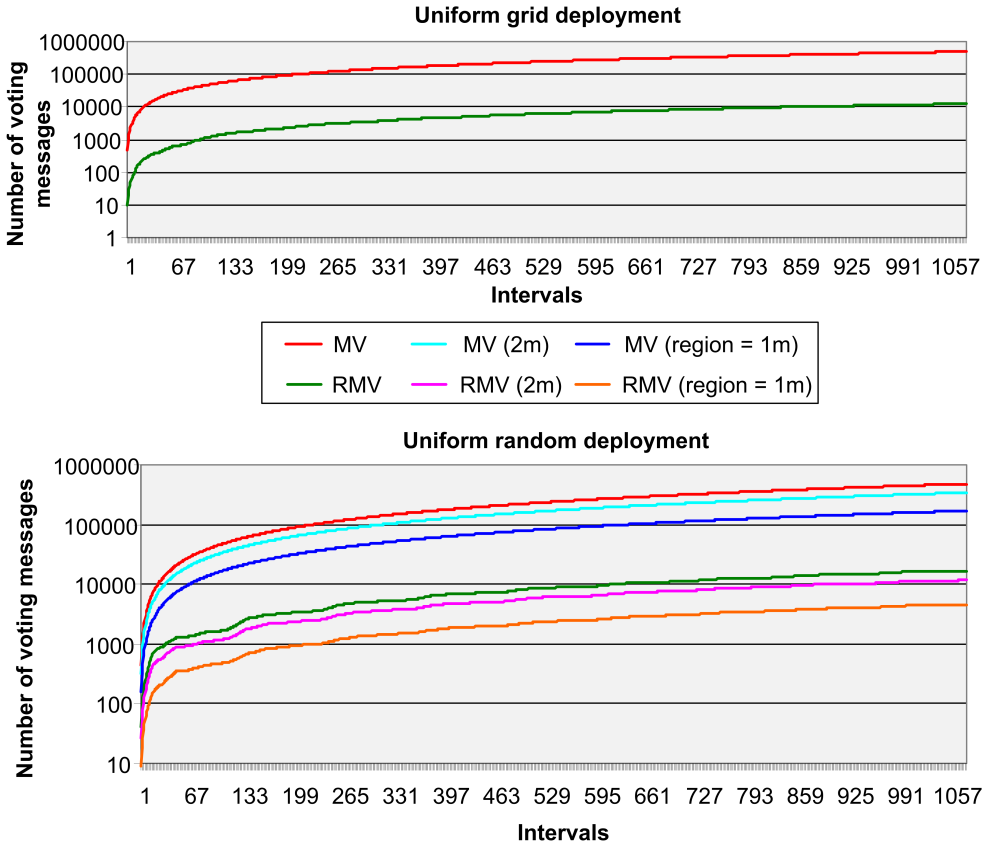


Figure B.10: Number of required voting messages applying MV and RMV in case of negative deviating sensor readings. The MV performs as usual requiring many voting messages. Despite the significantly lower detection performance, the RMV at least produces only marginal overhead by generating less messages.

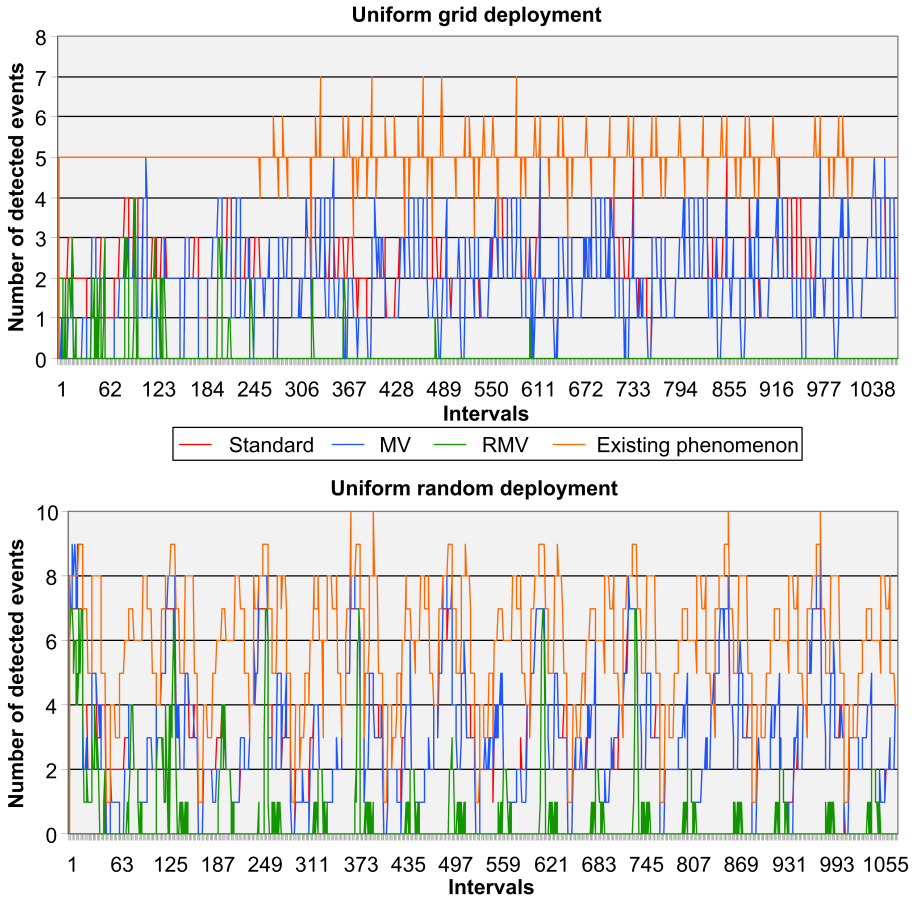


Figure B.11: Number of detected events per interval in case of negative deviating sensor readings. The standard detection is still able to detect the half of existing events and the results of MV closely meet the results of the standard method. In contrast to that, RMV performs bad and provides detection rates of 9% and 25% only.

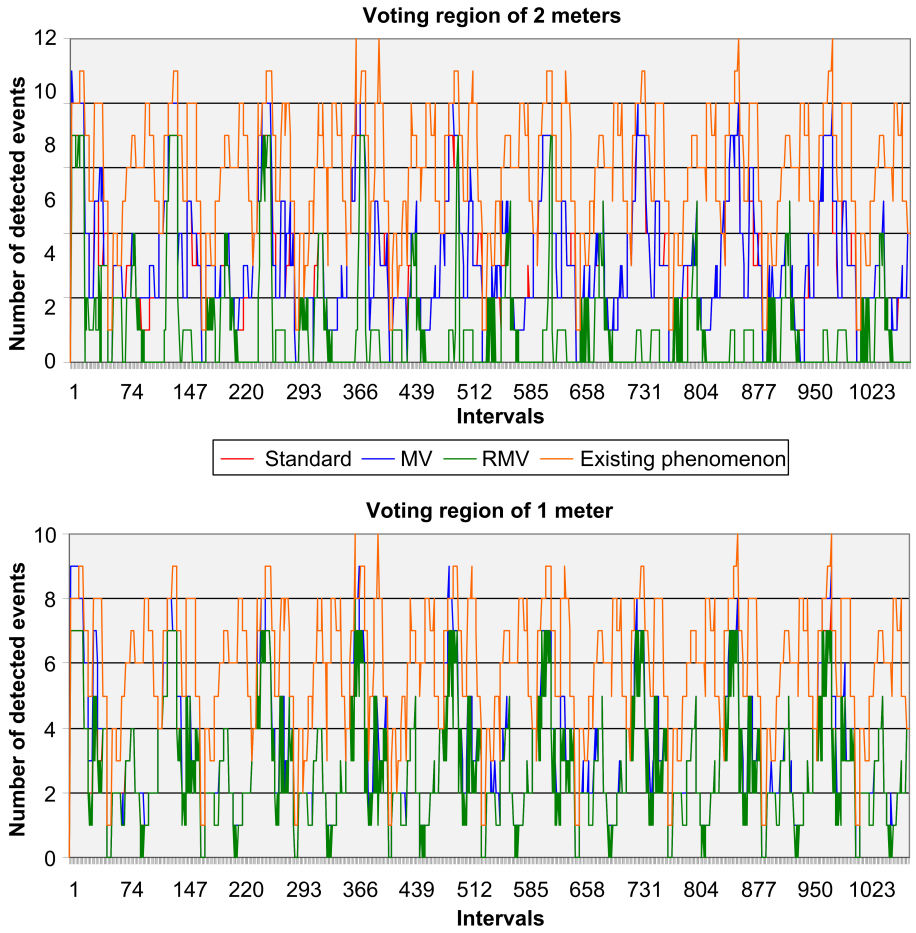


Figure B.12: Number of detected events per interval when applying different voting regions in case of negative deviating sensor readings. The standard detection and MV are still able to detect the half of existing events. Applying voting regions of 2 meters MV performs even better than the standard detection. The RMV performs bad and detects only about 40% of existing events even if a voting region of 1 meter is applied.

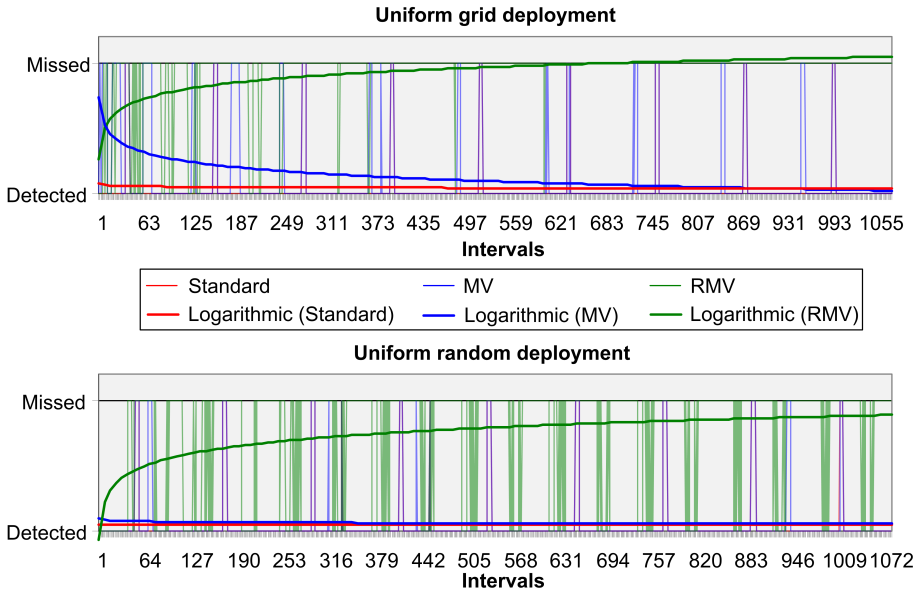


Figure B.13: Intervals with an undetected (missed) phenomenon. For better visualisation, the overall performance of all detection methods is represented by logarithmic trend curves. These trends represent the median rates of undetected phenomena. The performance of both voting approaches is unacceptable, but RMV performs significantly bad and misses 93 % of existing phenomena.

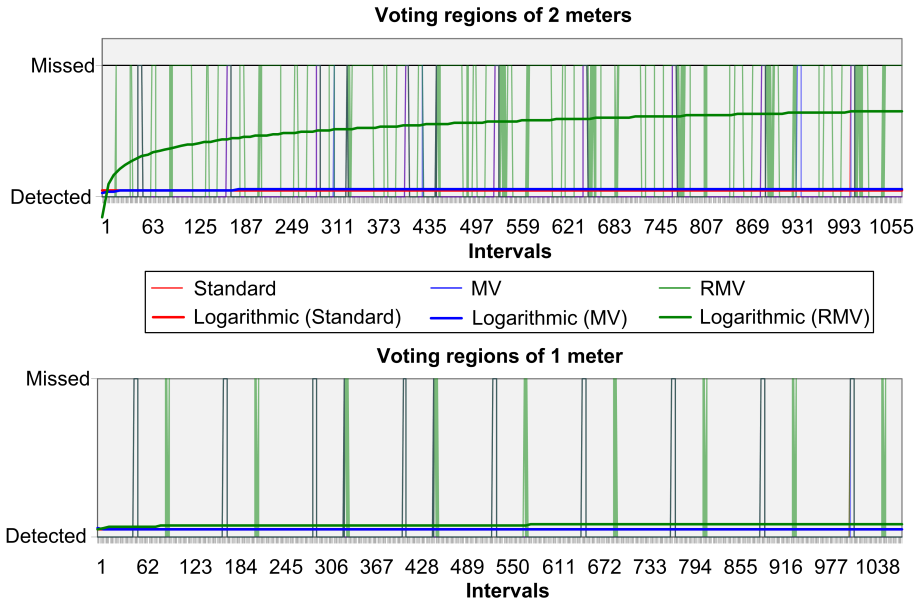


Figure B.14: Intervals with an undetected (missed) phenomenon. For better visualisation, the overall performance of all detection methods is represented by logarithmic trend curves. These trends represents the median rates of undetected phenomena. Downsizing the voting region reduces the number of missed phenomena. All approaches then converge to the standard detection. This is still unacceptable due to the overhead associated to voting.

B.3 Simulation results for general deviating sensor readings

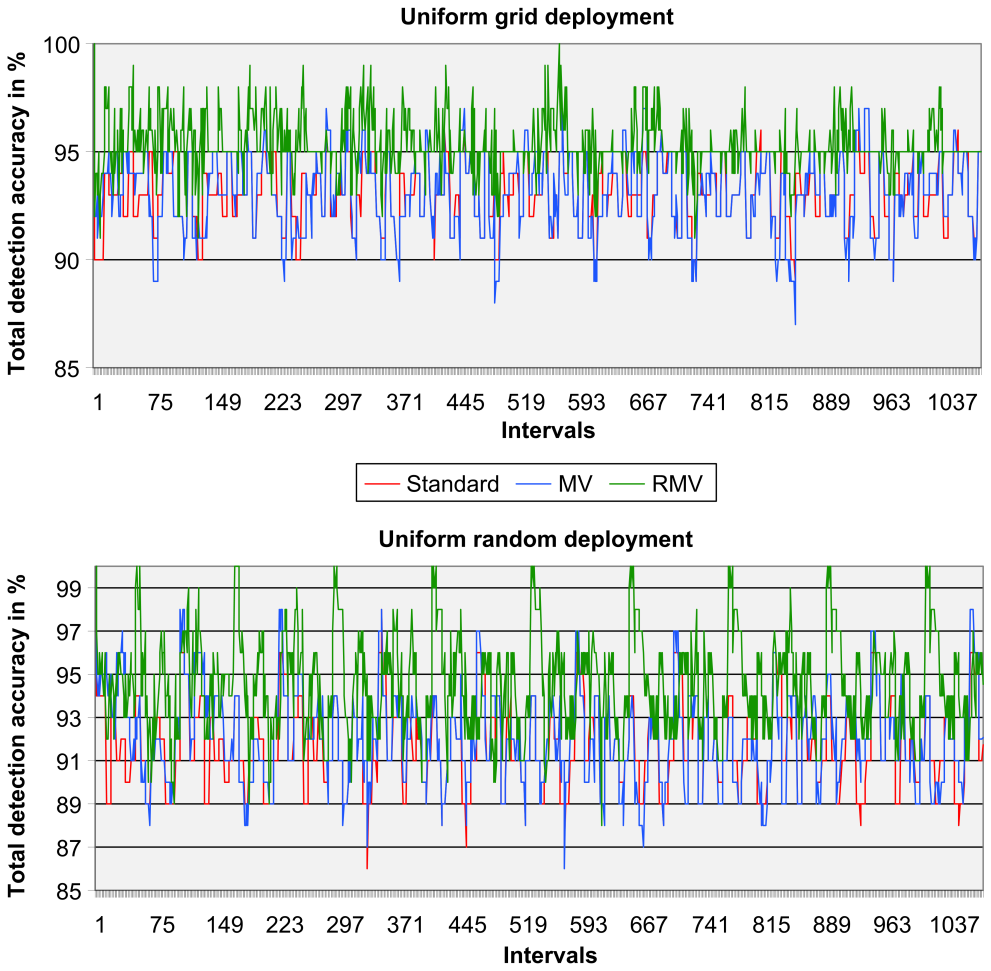


Figure B.15: Comparison of the total detection accuracy when applying MV and RMV in case of general deviating sensor readings. By reducing the number of *False positives*, the RMV approach enhances the accuracy of detection by about two to three percent. In contrast to that, the detection accuracy of MV is nearly equal to the standard detection accuracy.

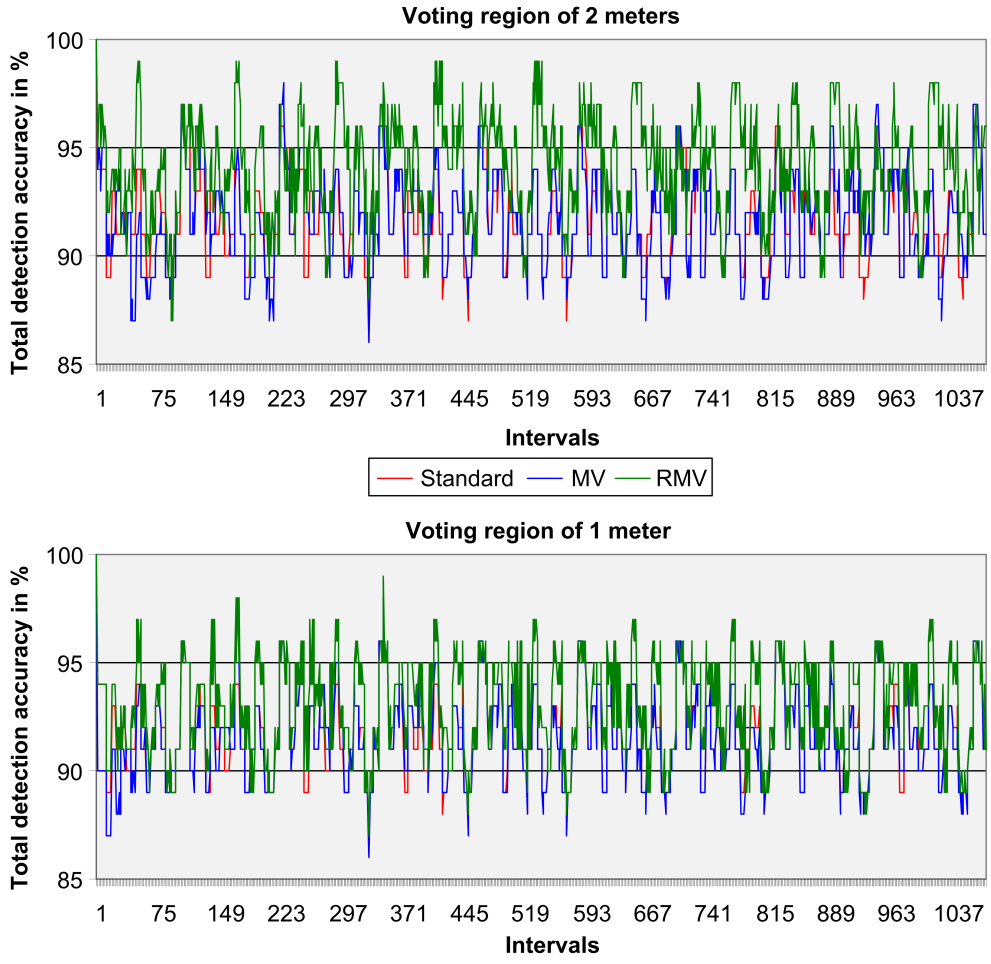


Figure B.16: Comparison of the total detection accuracy in application of different voting regions in case of general deviating sensor readings. Downsizing the voting region reduces the detection accuracy of both voting approaches.

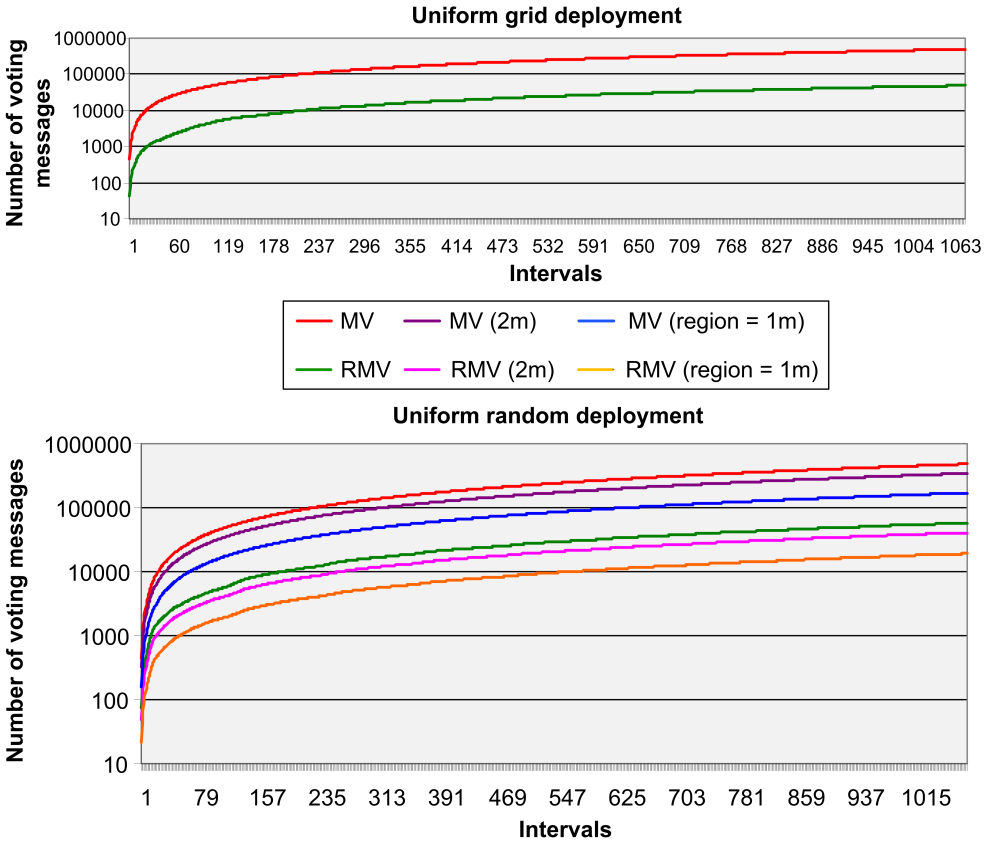


Figure B.17: Comparison of voting overhead in case of general deviating sensor readings. The RMV not only increases the detection accuracy but also requires significantly less messages than MV. In comparison to MV the RMV reduced the average number of messages by a factor of 9 using the standard voting region and a factor of 21 using a voting region of 2 meters.

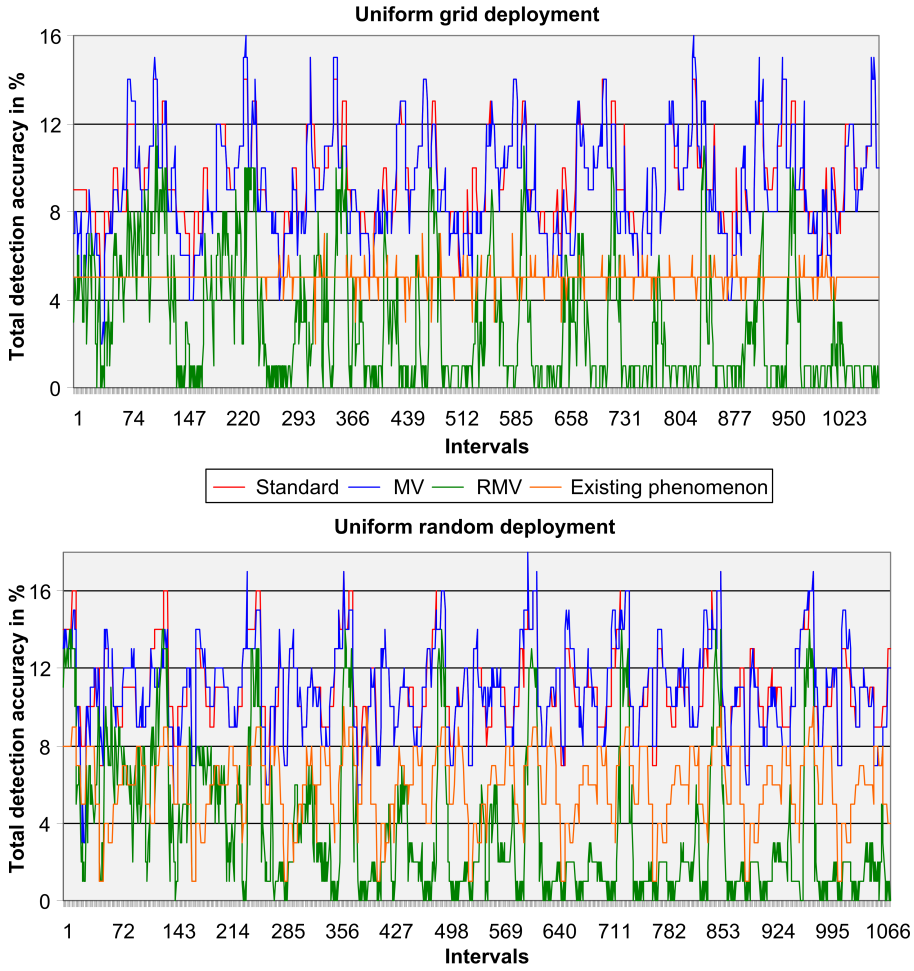


Figure B.18: Number of detected events per interval in case of general deviating sensor readings. The standard detection and MV almost double the number of detected events. In contrast to that, RMV performs contrary by missing almost half of all existing phenomena.

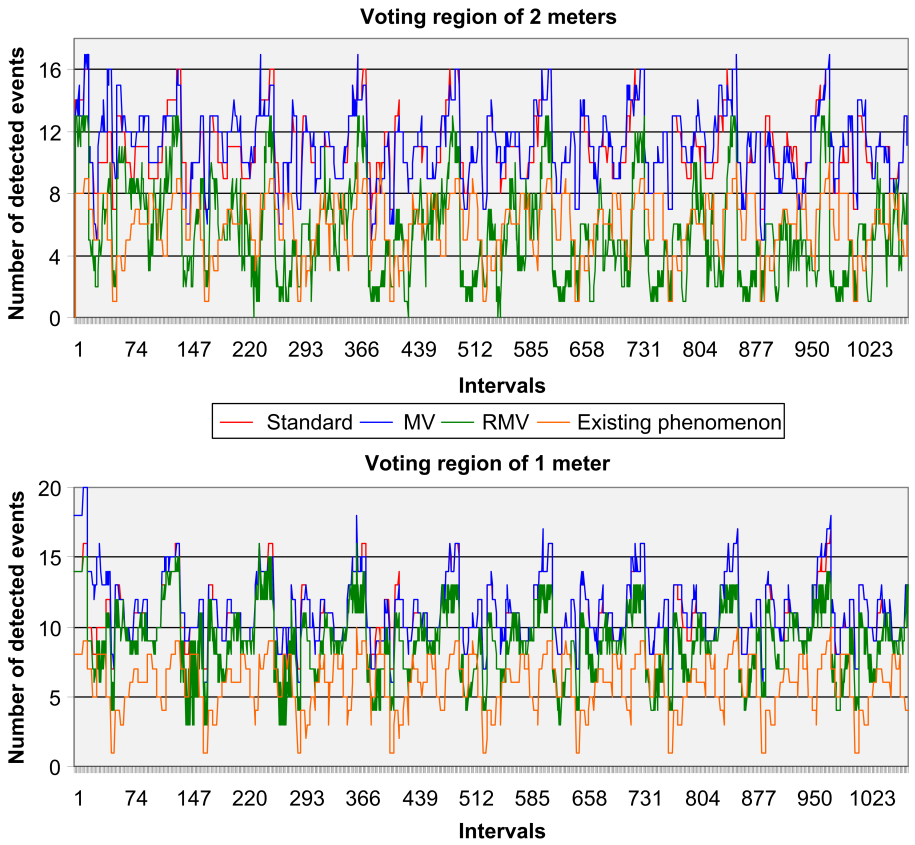


Figure B.19: Number of detected events per interval when applying different voting regions in case of general deviating sensor readings. In general, downsizing the voting region increases the number of detected events. Here, the RMV achieves by far the best results of all approaches (93% accuracy compared to the reference) by applying a voting region of 2 meters. Choosing a voting region of 1 meter is already too small for RMV and results in an increased number of *False positives*.

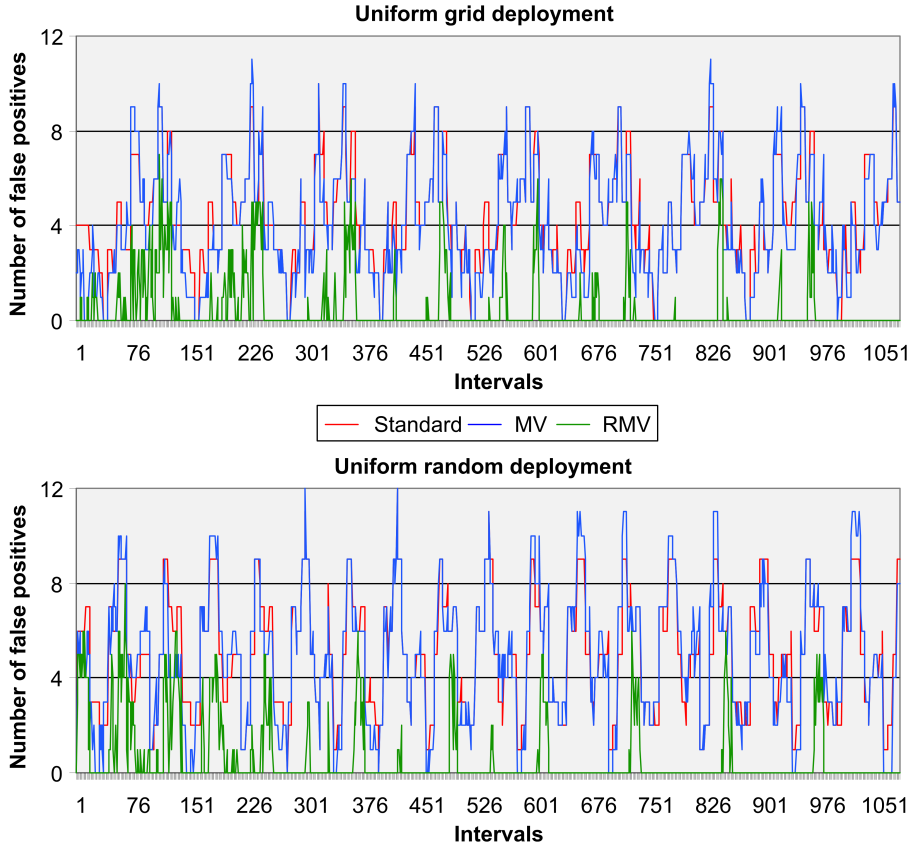


Figure B.20: Comparison of *False positives* per interval between the standard detection and voting. Here, MV and RMV apply a voting region of 2.5 meters. The standard detection and MV generate about 5% of *False positives*. Due to the low number of detected events, RMV also detects only few *False positives*.

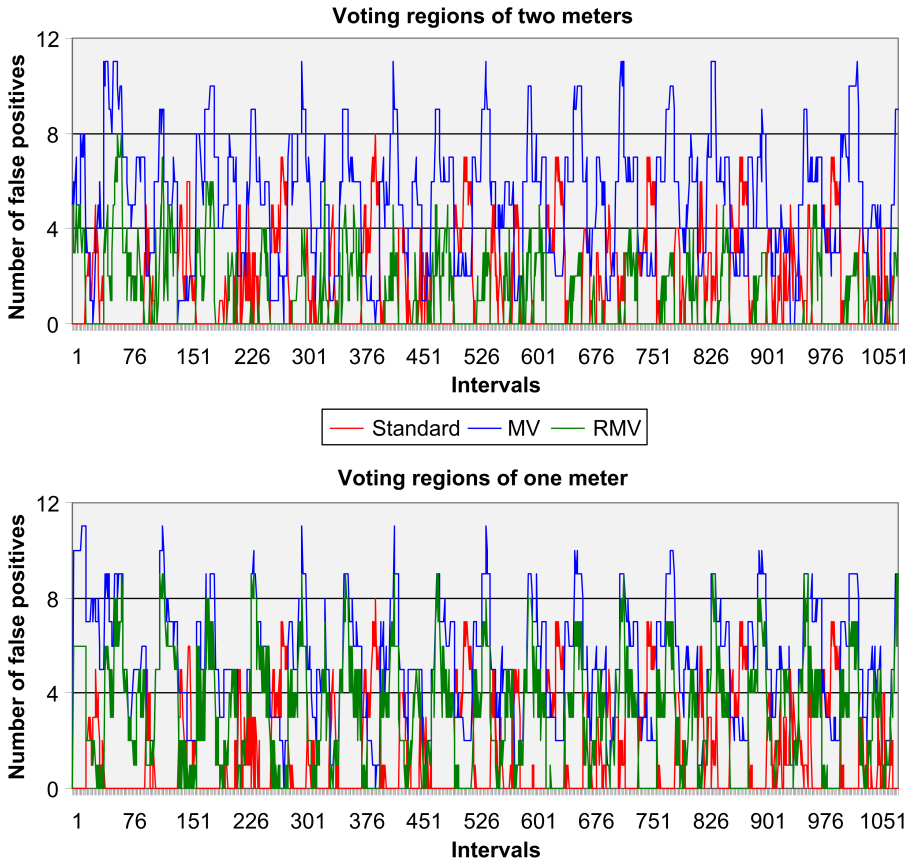


Figure B.21: Comparison of *False positives* per interval between the standard detection and voting. Here, MV and RMV apply voting regions of 2 meters and 1 meter. With downsizing the voting region the number of available voters decreases. This reduces the possibility that detected events are overruled by voting. Of course, the number of *False positives* increases as well. In application of a voting region of 2 meters, which is the best overall setting for RMV, RMV detects slightly more than one *False positive* in average.

B.4 Simulation results for permanently failing sensing capabilities

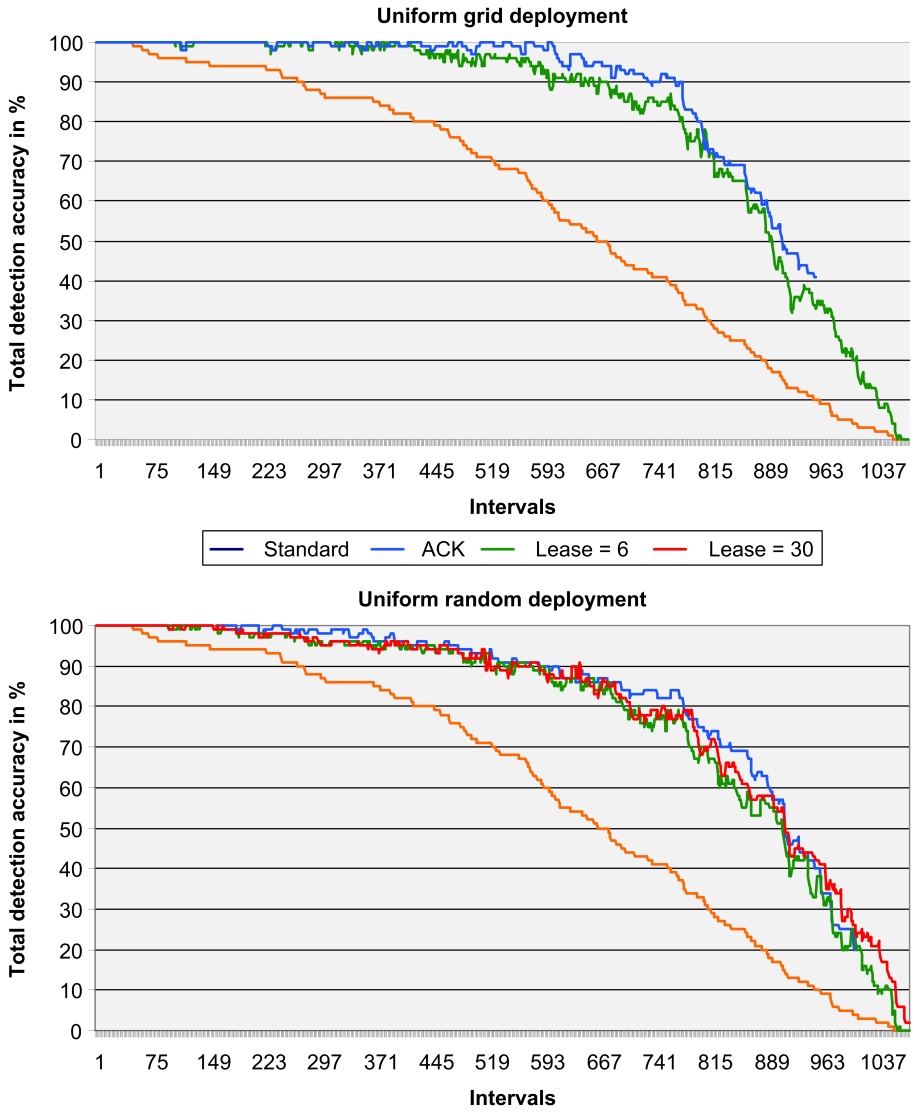


Figure B.22: Comparison of detection results when applying lease-based and ACK-based collaboration in case of permanent failing sensing capabilities. In both deployments, the lease-based approach enhances the detection accuracy by 30% to 35%. The ACK-based scheme indicated similar or even better performance but has been aborted by the simulation environment due the huge number of messages needed. Moreover, even if only 50% of all nodes are able to generate local detection results in the standard scheme, both collaboration schemes still provide an detection accuracy that is higher than 80%.

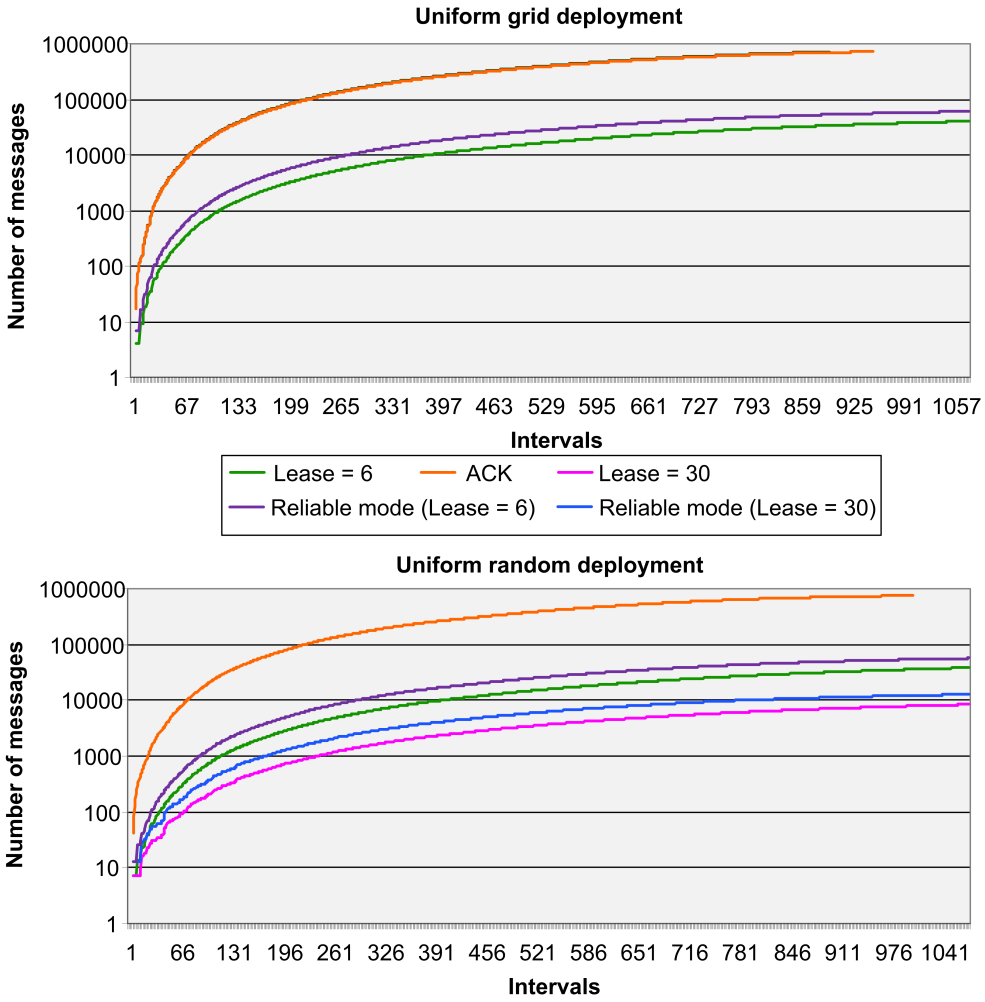


Figure B.23: Comparison of required messages in the entire network in application of lease-based and ACK-based collaboration in case of permanent failing sensing capabilities. Despite of the significantly enhanced detection performance, the lease-based approach only requires to transmit 0.35 messages per node and interval. Also using the reliable mode in lease-based collaboration, which does not influence the detection results, requires to transmit about one message in two intervals. In contrast to that, the ACK-based approach required in average more than 7 messages per node and interval. This caused the simulation runs to be aborted by the simulator. Please note, the diagrams applied logarithmic scales.

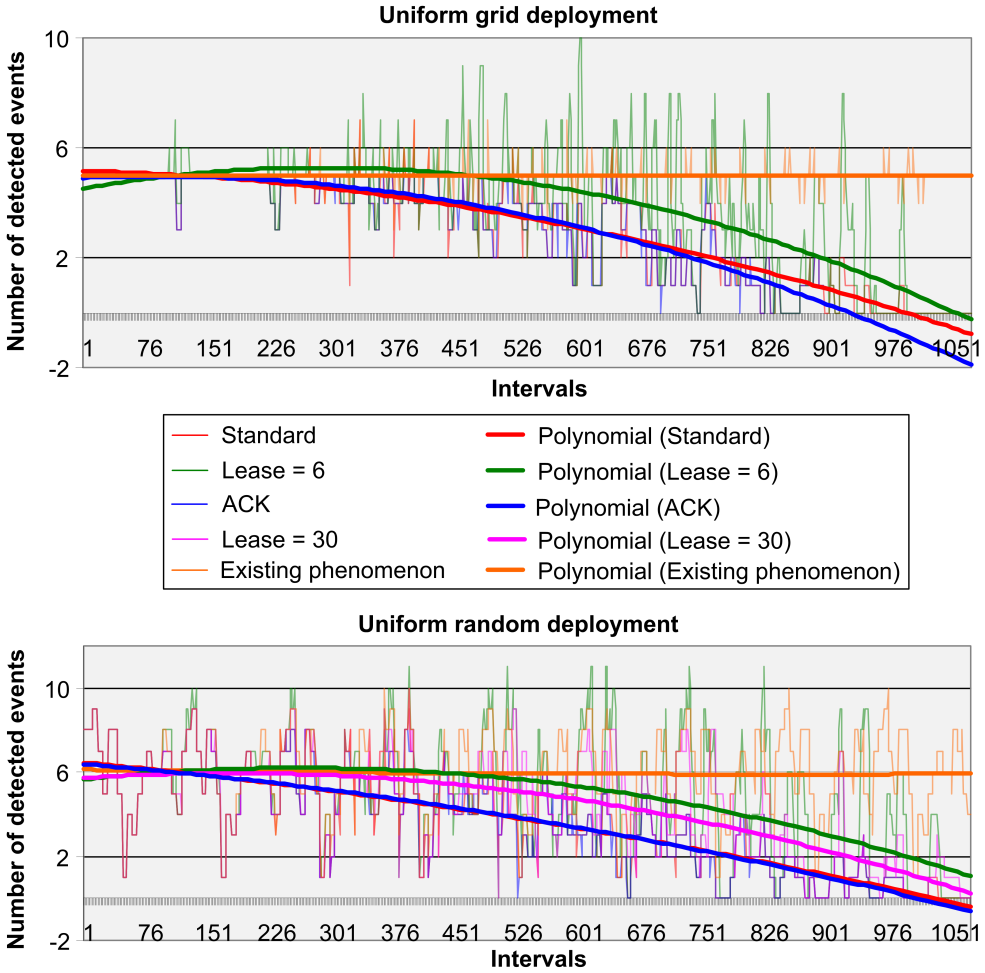


Figure B.24: Comparison of detected events applying the lease-based and ACK-based collaboration schemes in case of permanent failing sensing capabilities. For better visualisation, the overall performance of all detection methods is represented by polynomial trend curves. These trends represents the median rates of detected *False positives*. The lease-based approach significantly improves the standard detection results by 28% (grid) and 39% (random). In average, it detects 77% (grid) and 82% (random) of all existing events. The ACK-based approach performs well until its abort, which results in a negative trend.

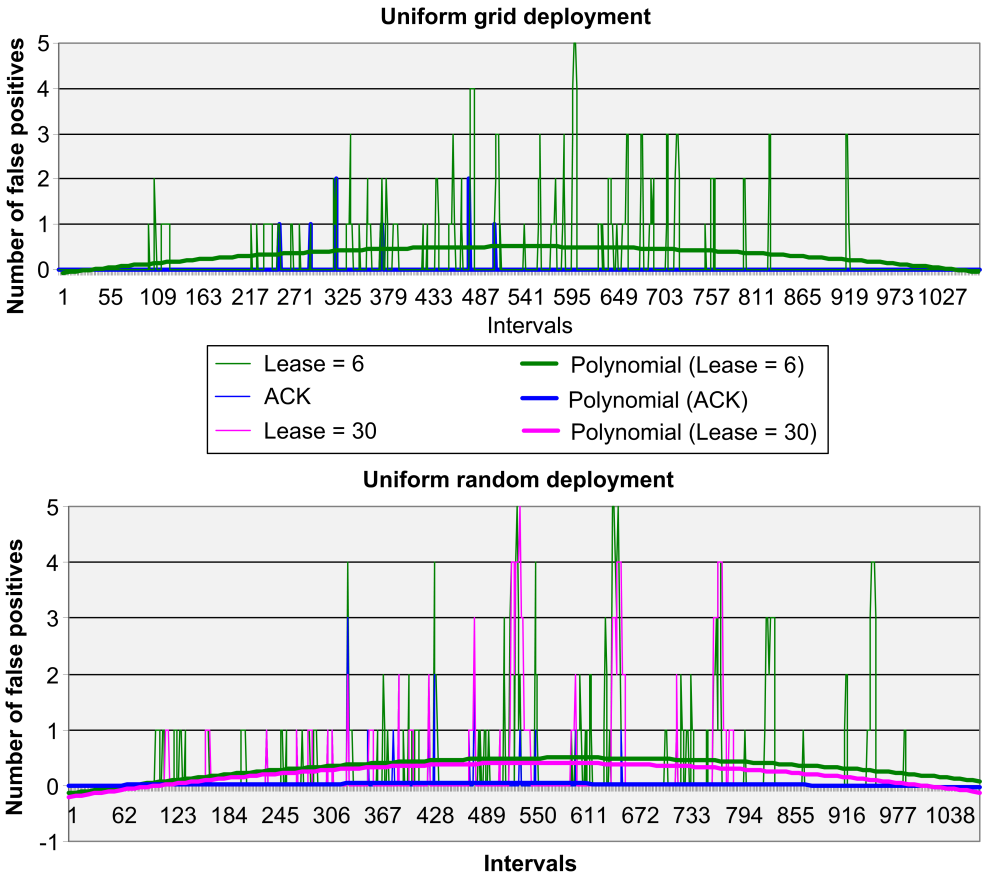


Figure B.25: Comparison of detected *False positives*. For better visualisation, the overall performance of all detection methods is represented by polynomial trend curves. These trends represents the median rates of detected *False positives*. Both approaches clearly detect only few *False positives*. Due to the fact, that sensor nodes either correctly detect an event or fail completely, the standard performance features no *False positives*.

Figure B.26: Comparison of undetected events in case of permanently failing sensing capabilities. Again, polynomial trend curves represent the median rates of undetected event for each approach. Here, the total number of undetected events is depicted. The trend curves clearly indicate that the lease-based schemes outperform all other approaches.

B.5 Simulation results for transiently failing sensing capabilities

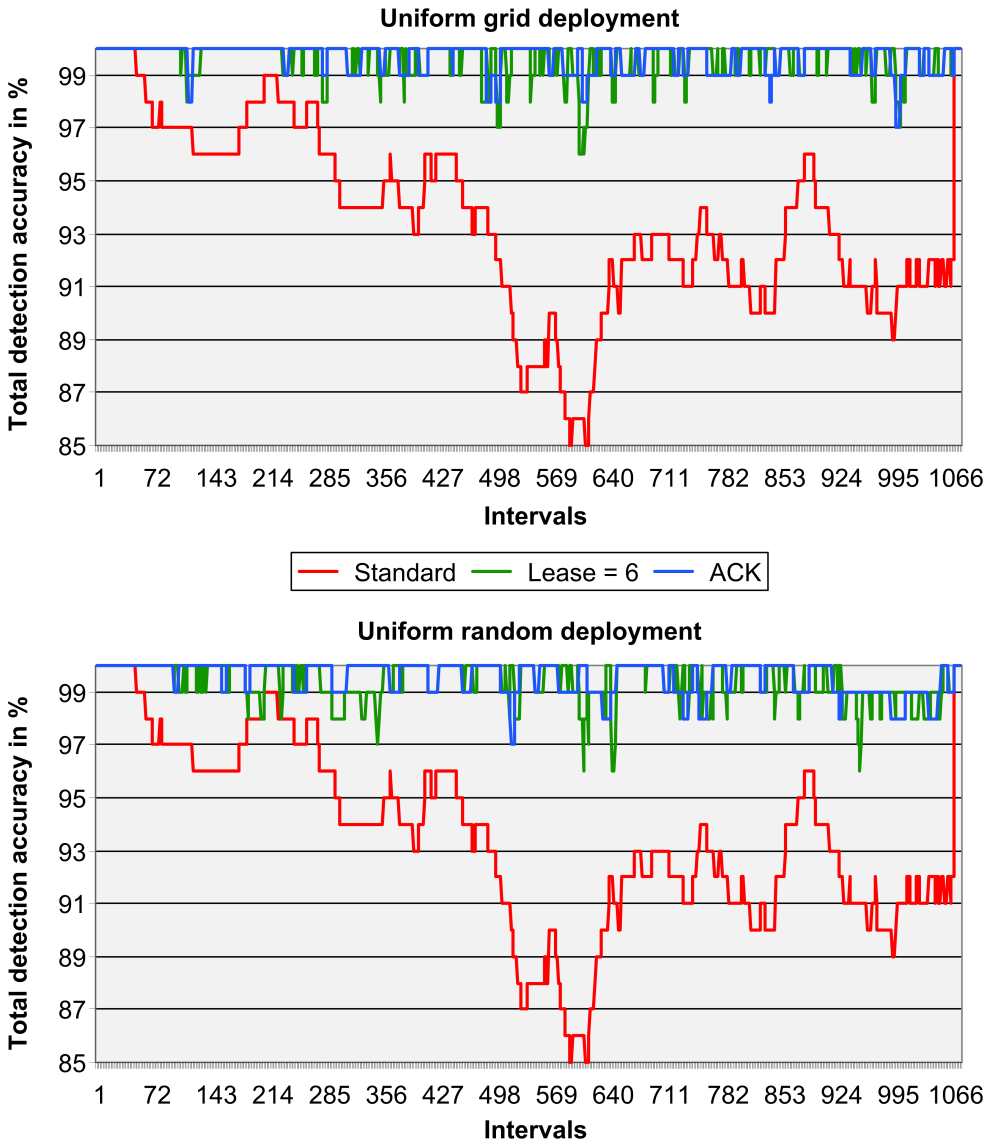


Figure B.27: Comparison of detection results when applying lease-based and ACK-based collaboration in case of transiently failing sensing capabilities. Both collaboration methods perform excellent and feature a detection accuracy of nearly 100% . In average, that enhances the standard detection by 6%.

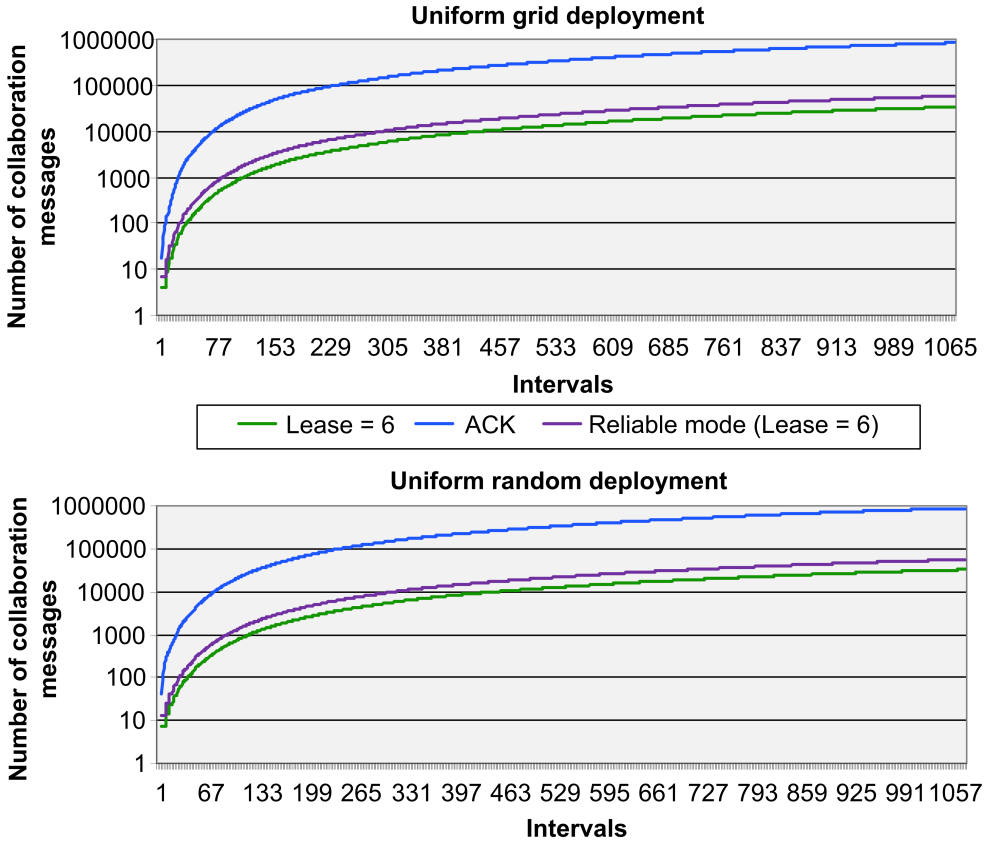


Figure B.28: Comparison of required messages in application of lease-based and ACK-based collaboration in case of transiently failing sensing capabilities. The lease-based approach only requires to transmit 0.3 messages per node and interval whereas the number of ACK-based messages are multiplied by at least a of factor 25.

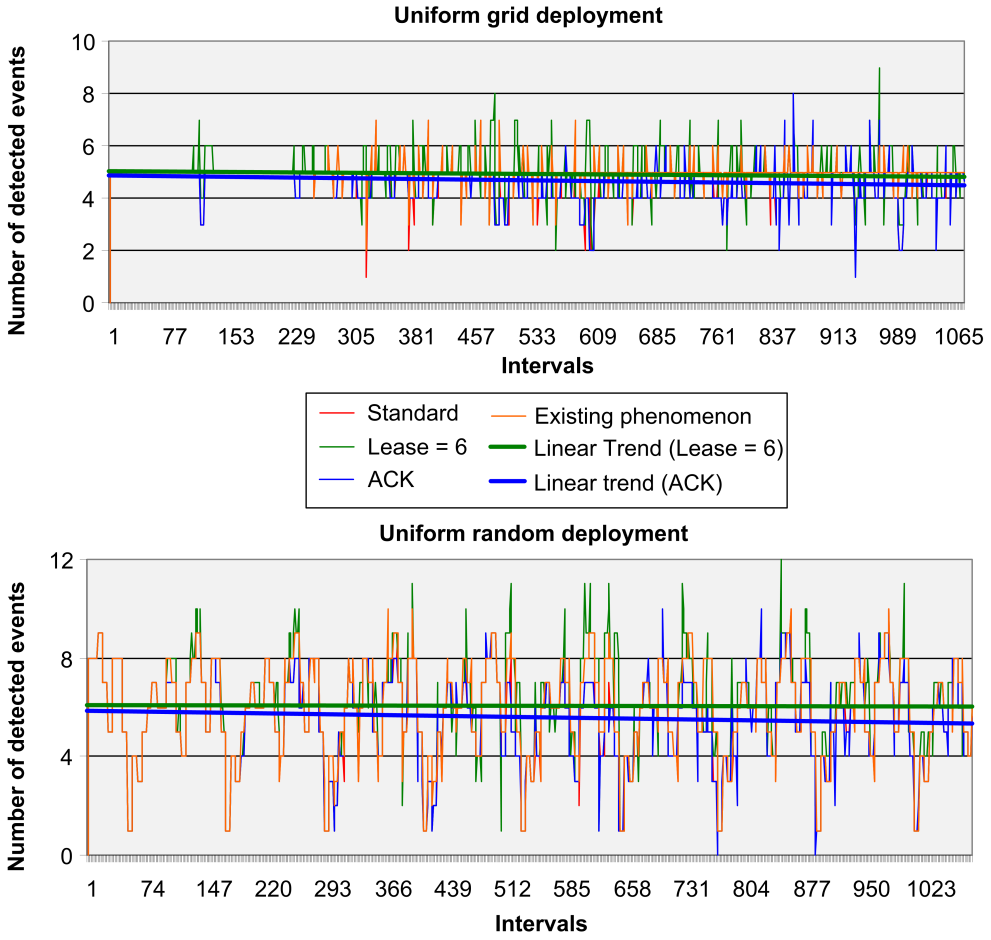


Figure B.29: Comparison of detected events applying the lease-based and ACK-based collaboration schemes in case of transiently failing sensing capabilities. In comparison to the reference, the lease-based approach features a deviation of only 1.5% whereas both other methods deviate by about 7%. In general, the number of detected results is less affected by the transient failure. For details refer to the failure scenario.

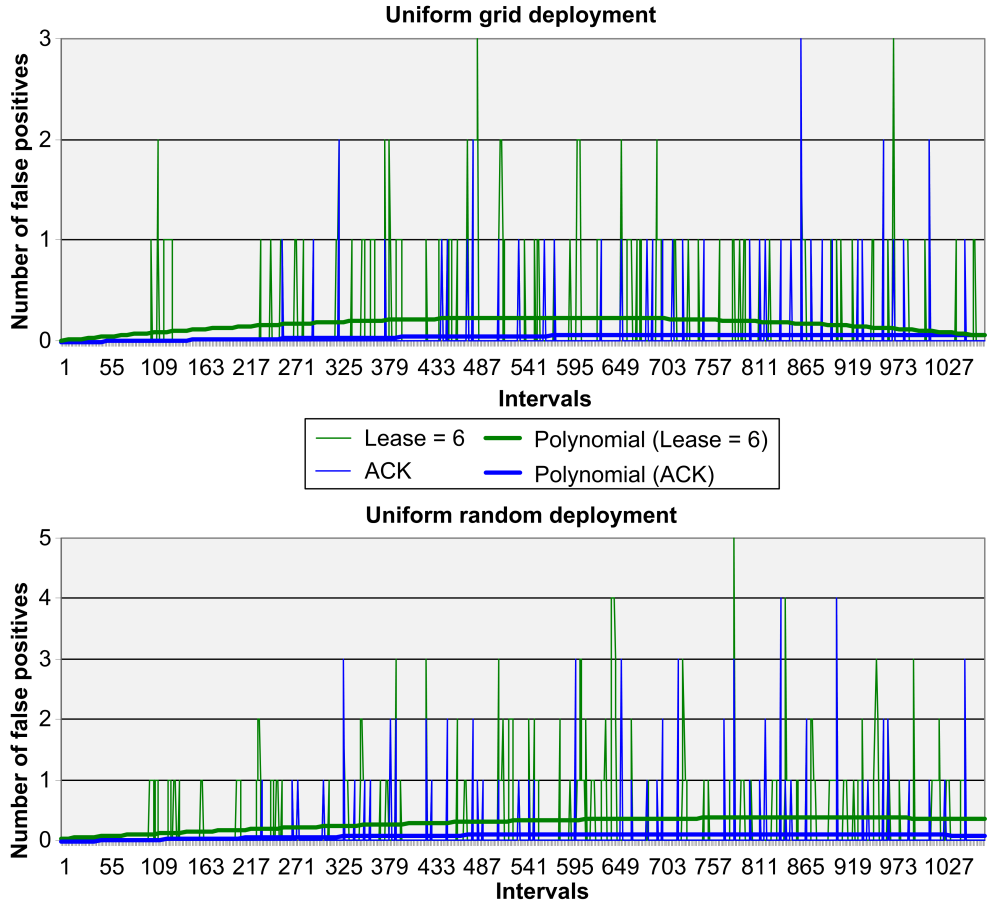


Figure B.30: Comparison of detected *False positives*. For better visualisation, the overall performance of all detection methods is represented by polynomial trend curves. These trends represents the median rates of detected *False positives*. Both approaches clearly detect only few *False positives*. Due to the fact, that sensor nodes either correctly detect an event or fail completely, the standard performance features no *False positives*.

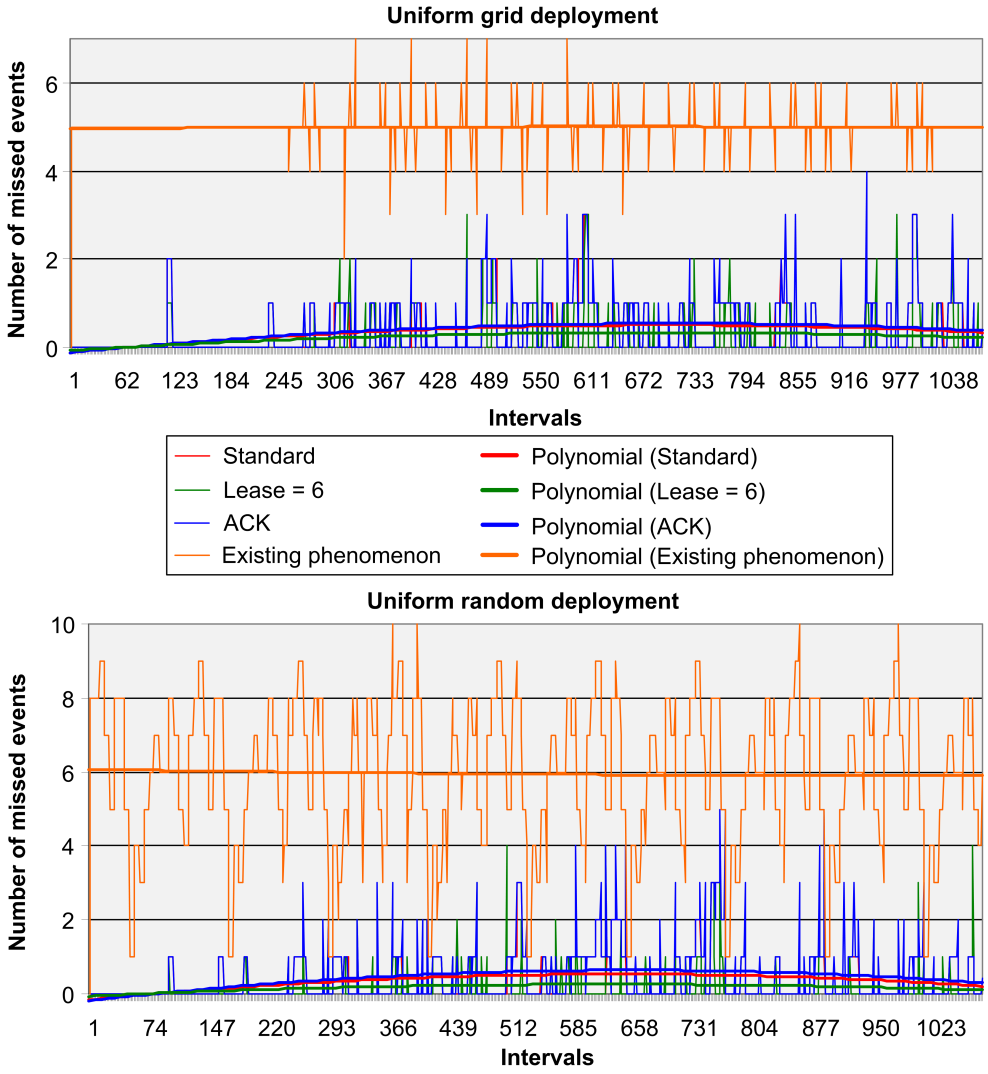


Figure B.31: Comparison of detected *False positives*. The number of undetected events is rather low. The polynomial trend curves show a similar number of undetected event for the standard detection and the ACK-based scheme. The rate of undetected events is even lower in the lease-based scheme. However, all detection method have not missed a phenomenon.

B.6 Simulation results for simultaneous occurrence of deviations and transient failures

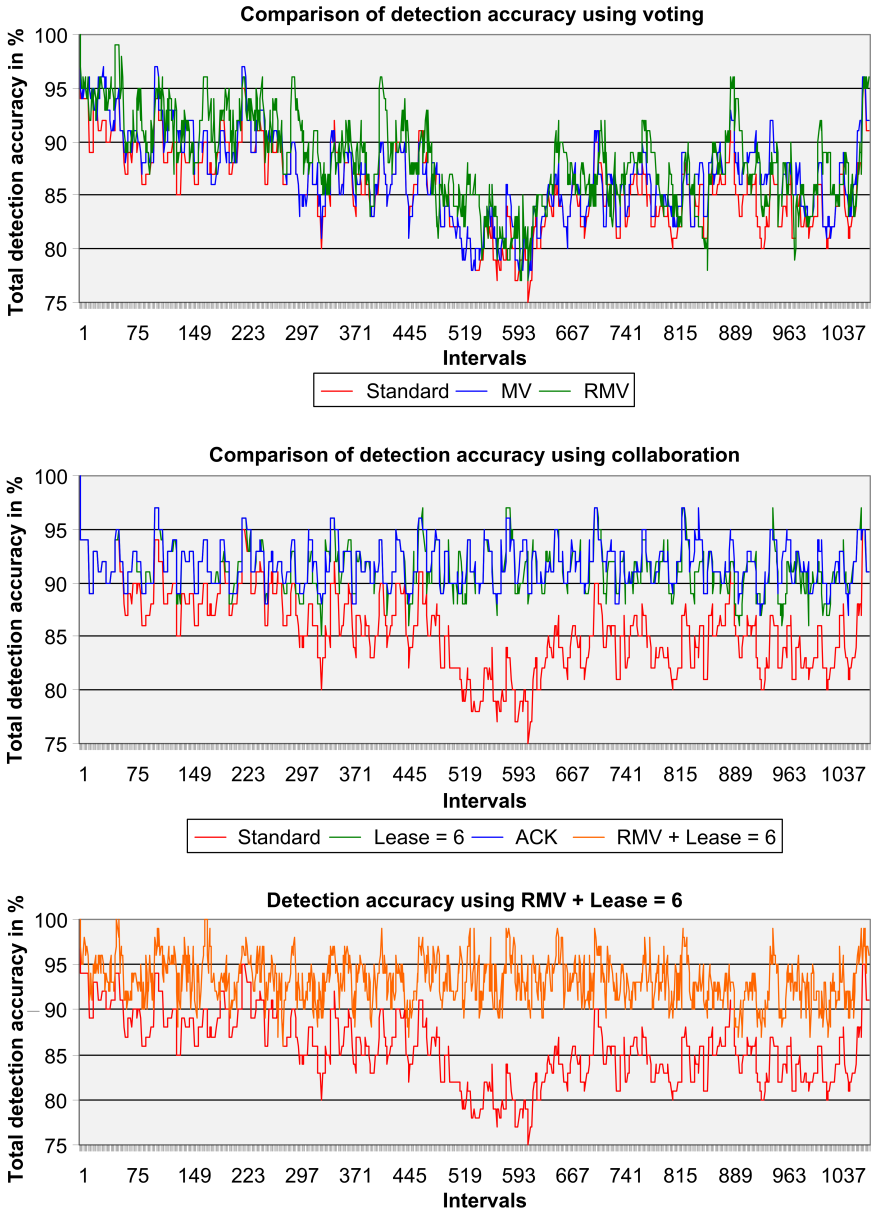


Figure B.32: Comparison of detection accuracy of all introduced detection methods in case of general deviations and transiently failing sensing capabilities. The collaboration methods clearly perform better than the standard detection and the voting approaches. A combination of RMV and lease-based detection provided the best results with a total detection accuracy of 93%.

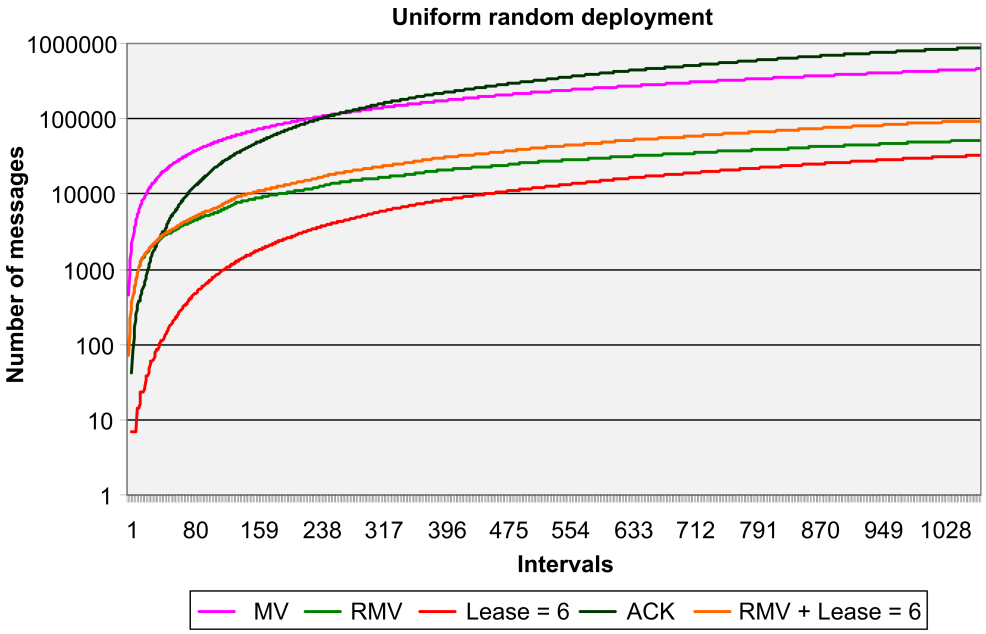


Figure B.33: Comparison of required messages in application of all introduced detection methods in case of general deviations and transiently failing sensing capabilities. The RMV and the lease-based detection performed best. A combination of both methods provides the best detection accuracy while the associated overhead of less than one message per node and interval is still acceptable.

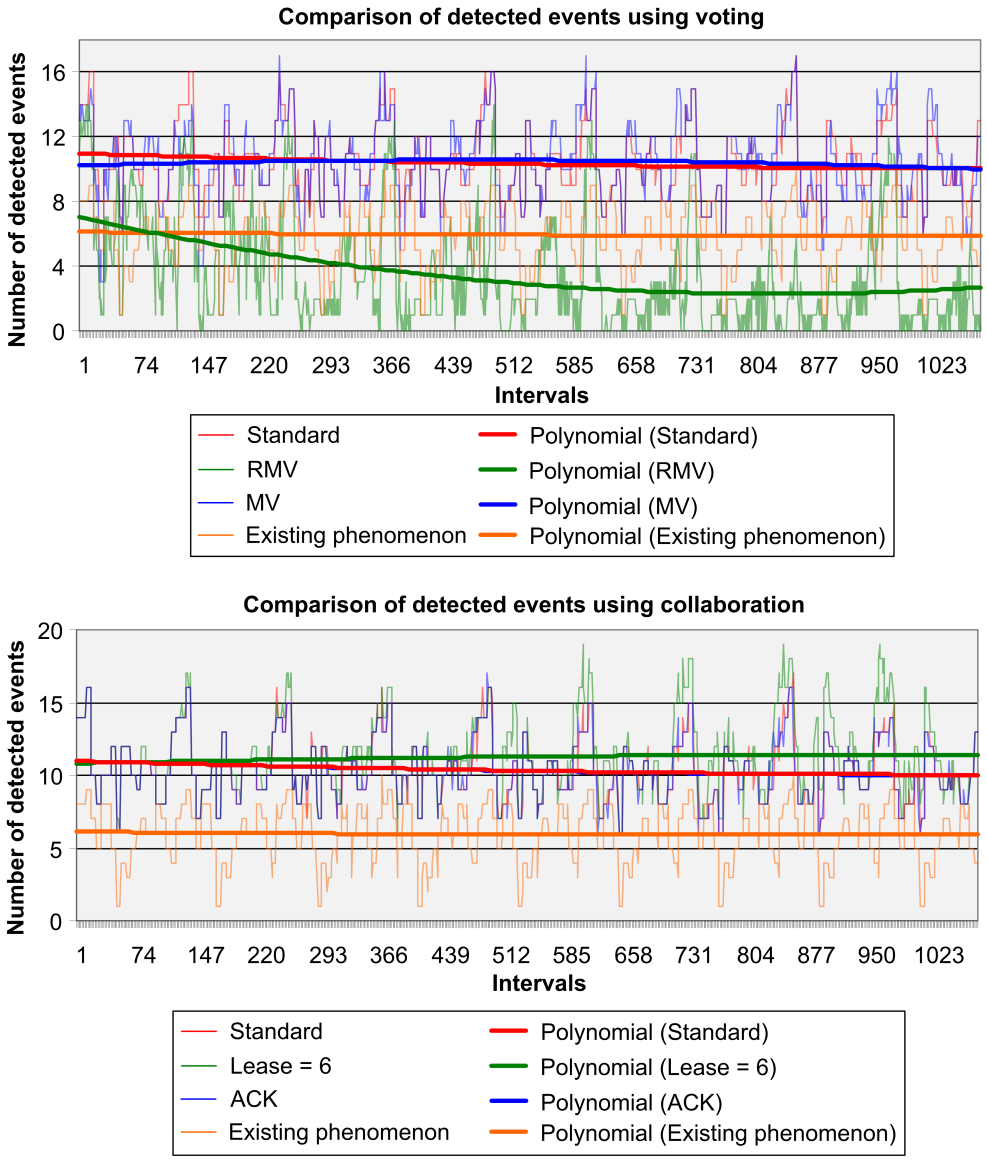


Figure B.34: Comparison of detected events in case of general deviations and transiently failing sensing capabilities. Only the RMV shows a significant trend to detect less events than phenomena existed. This is caused by overruling of detected events. All other approaches clearly tend to detect more events than phenomena existed. These events are caused by deviating sensor readings.

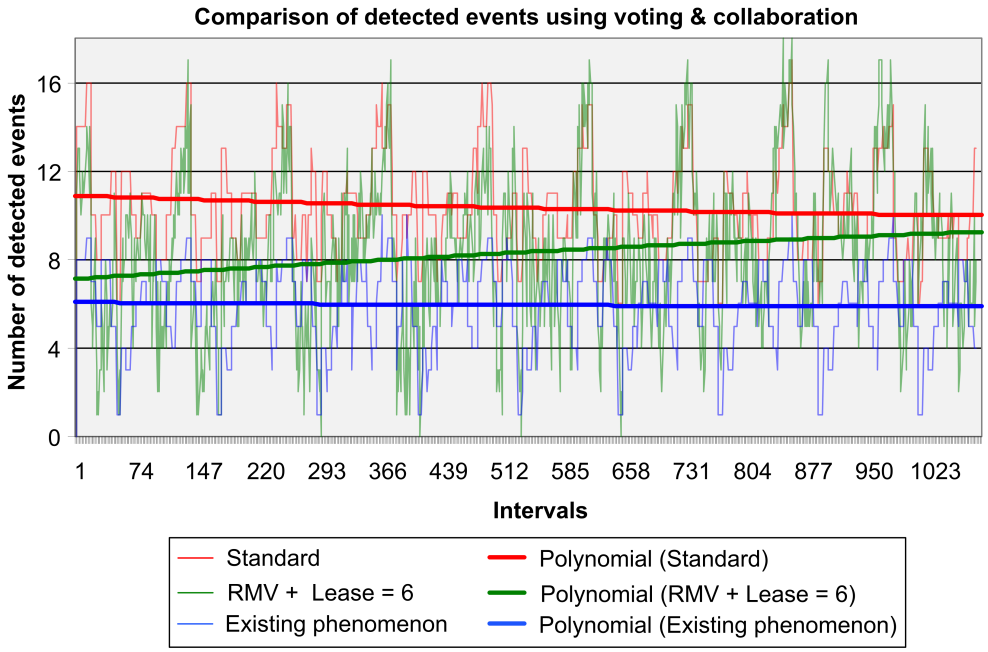


Figure B.35: Number of detected events in case of general deviations and transiently failing sensing capabilities using a combination of lease-based detection and RMV. Both approaches well completed each other. The RMV reduces the number of *False positives* caused by the lease-based collaboration. On the other hand, the lease-based collaboration enabled to continue event detection in case of lost sensing devices.

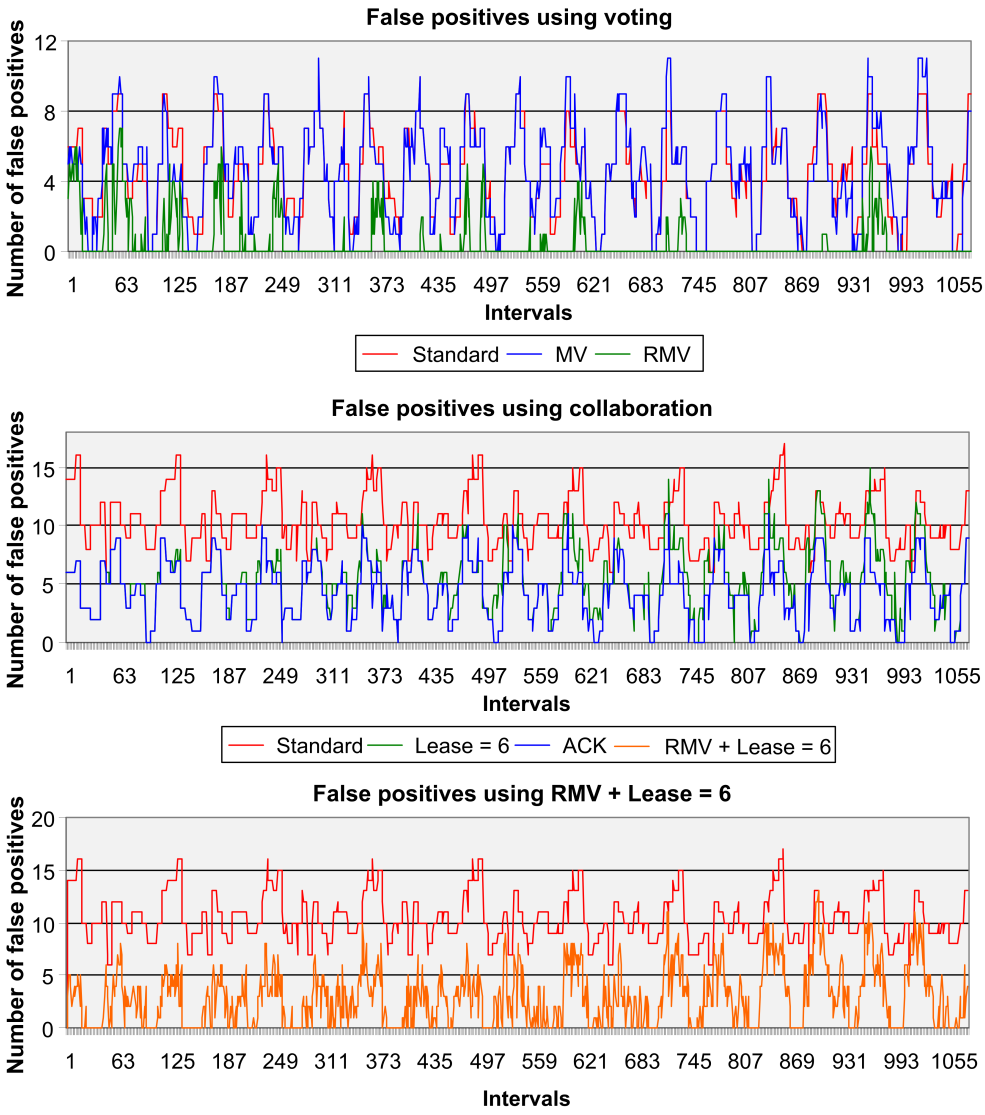


Figure B.36: Comparison of detected *False positives* in case of general deviations and transiently failing sensing capabilities. Despite of RMV, all approaches detected more events than phenomena existed caused by deviating sensor readings. This in turn increases the number of *False positives* in these approaches.

List of Symbols and Abbreviations

AAL Ambient Assisted Living
ACK acknowledgement
BAN Body Area Network
CoDED Context Dependent Event Detection
COUGAR The Sensor Network is the Database
CSS Chirp Spread Spectrum
CWV Confidence Weighted Voting
DHCP Dynamic Host Configuration Protocol
DWV Distance Weighted Voting
EDF Earliest Deadline First
EDT Event Decision Tree
ESL Event Specification Language
FIFO First In First Out
GFSM Generating Finite State Machine
GPS Global Positioning System
GUI Graphical User Interface
IP Internet Protocol
JVM Java Virtual Machine
MAC Medium Access Control
MV Majority Voting
NIST National Institute of Standards and Technology
OS Operating System
P2P Peer-to-Peer
QoI Quality of Information
QoS Quality of Service
REFLEX Realtime Event FLow EXecutive
RMV Reactive Majority Voting
RSSI Received Signal Strength Indication

SPoF Single Point of Failure
STOP SpaceTime Oriented Programming
SQL Structured Query Language
TASK Tiny Application Sensor Kit
TinyDB Declarative Database for Sensor Networks
TinyOS Tiny Operating System
WSN Wireless Sensor Network
XML eXtensible Markup Language

List of Figures

List of Tables

Listings